Hierarchical Symbolic Analysis of Analog Integrated Circuits via Determinant Decision Diagrams

Xiang-Dong Tan, *Member, IEEE* and C.-J. Richard Shi, *Senior Member, IEEE*

***Abstract*—A new method is proposed for hierarchical symbolic analysis of large analog integrated circuits. It consists of performing symbolic suppression of each subcircuit to its terminals in terms of subcircuit matrix determinants and cofactors, and applying Cramer's rule to symbolically solve the set of equations at the top level of the circuit hierarchy. An annotated, directed, and acyclic graph, called determinant decision diagram (DDD), is used to represent symbolic determinants of subcircuit matrices and cofactors used in subcircuit suppression, as well as symbolic determinants of the top-level circuit matrix and cofactors required in applying Cramer's rule. DDD enables us to systematically exploit the inherent sparsity of circuit matrices and the sharing of symbolic expressions. It is capable of representing a huge number of symbolic product terms in a canonical and highly compact manner. The proposed method is illustrated using a Cauer parameter low-pass filter. It has been implemented in a symbolic analyzer and compared to best-known hierarchical symbolic analyzer SCAPP and numerical simulator SPICE. Experimental results on several analog circuits including the $\mu A741$ operational amplifier—a circuit with less structural regularities—are described.**

***Index Terms*—Analog circuit design, analog symbolic analysis, determinant decision diagrams, hierarchical analysis.**

I. Introduction

SYMBOLIC analysis calculates the behavior or the characteristic of a circuit in terms of symbolic parameters. It is important for many circuit-design applications such as optimum topology selection, design space exploration, behavioral model generation, and fault detection [5]. Symbolic analysis, however, has not been widely used by circuit designers. The root of the difficulty is apparent: the number of product terms in a fully-expended symbolic expression may increase exponentially with the size of a circuit. Any manipulation and evaluation of symbolic expressions would require CPU time at best linear in the number of terms and, therefore, have both the time and space complexities exponential in the size of a circuit.

To cope with the circuit-size limitation problem, modern symbolic analyzers rely on two techniques: symbolic simplification and hierarchical decomposition [3]. Symbolic simplification discards those insignificant terms based on the relative numerical magnitudes 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], [7], [13], [21] or after the generation [2], [4], [20]. 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 computer-aided circuit optimization and testability analysis.

Hierarchical decomposition generates circuit transfer functions as either nested symbolic expressions or sequences of symbolic expressions. There are three methods known as topological analysis [14], network formulation [6], and two-port decomposition [8].

- *Topological Analysis* [14]: The circuit topology is represented as a directed graph and circuit parameters are represented as the weights of the edges in the graph. Hierarchical decomposition is carried out on the directed graph. Subcircuit analysis amounts to finding node-disjoint directed paths and node-disjointed directed loops. Results obtained in subcircuit analysis are combined upward until the root circuit is reached.
- *Network Formulation* [6]: Hierarchical decomposition is performed directly on the system equations. The decomposition procedure is characterized by eliminating variables one at a time (called reduced modified nodal analysis) for each subcircuit analysis. The results of lower-level subcircuits are combined according to some rules to form the equation sets for upper-level subcircuits. The process continues until the root circuit is reached, and the transfer function is computed from the resulting equation set. All the intermediate steps are expressed as a *sequence of expressions*.
- *Two-Port Decomposition* [8]: Two-port decomposition derives the equivalent circuit for each subcircuit based on the two-port circuit theory, and then uses the equivalent circuit to perform hierarchical symbolic analysis. The two-port decomposition method is essentially a generalization of the network formulation method [6] where types of terminal variables can be selected (currents or voltages) based on the types of circuit ports that a designer chooses.

Unfortunately, no systematic mechanisms exist to fully exploit the expression sharing and circuit sparsity in existing hierarchical symbolic analysis methods. The resulting expressions are not compact enough. Manipulations, other than evaluation, of

Manuscript received February 28, 1999; revised October 27, 1999. This work was sponsored by the U.S Defense Advanced Research Projects Agency (DARPA) under Grant F33615-96-1-5601, from the United States Air Force, Wright Laboratory, Manufacturing Technology Directorate, and by Conexant Systems, Inc. Some preliminary results of this paper appeared in *Proc. IEEE Int. Symp. Circuits and Systems*, 1998, Monterey, CA, May 31–June 3, 1998. This paper was recommended by Associate Editor K. Mayaram.

X.-D. Tan was with the Department of Electrical Engineering, University of Washington, Seattle, WA 98195 USA. He is now with Monterey Design Systems, Sunnyvale, CA 94089 USA.

C.-J. R. Shi is with the Department of Electrical Engineering, University of Washington, Seattle, WA 98195 USA (e-mail: cjshi@ee.washington.edu).

Publisher Item Identifier S 0278-0070(00)03220-6.

the resulting sequences of expressions are known to be complicated and often require dedicated efforts, e.g., sensitivity calculation in [10] and lazy approximation in [13].

In this paper, we present a new hierarchical method for exact symbolic analysis. It takes advantage of both hierarchical decomposition and a recently introduced graphical representation of symbolic determinants called determinant decision diagrams (DDD's) [11], [12]. DDD's can exploit the *sparsity* of circuit matrices and the *sharing* among symbolic expressions in a systematic manner. For example, the determinant of the circuit matrix of an n-section ladder circuit can be represented by a DDD with $3n-2$ vertices, which represents $F(n+1)$ product terms, where $F(i)$ is the ith Fibonacci number [11], [12]. For a 30-section ladder circuit, over 1.3 million product terms can be represented by a DDD with only 88 vertices. In the worst case, the number of DDD vertices (called the DDD size) can grow exponentially with the size of a circuit. Fortunately, for practical analog circuits, the number of DDD vertices are generally many orders of magnitude less than the number of product terms. More importantly, manipulations such as cofactoring and sensitivity can be performed in almost linear time in the size of a DDD.

The rest of the paper is organized as follows. Following an overview of the general procedure for hierarchical circuit analysis in Section II, Section III presents the basic idea underlying the proposed DDD-based hierarchical symbolic analysis method. Section IV reviews the concept of DDD's. Section V illustrates the application of the proposed method to an analog circuit. The complete algorithm is summarized in Section VI. Section VII describes experimental results and the comparison to symbolic analyzer SCAPP and numerical simulator SPICE on several practical analog circuits. Section VIII concludes the paper.

II. Overview of Hierarchical Circuit Analysis

For a linear(ized), time-invariant analog circuit, its system of equations can be formulated by, for example, the modified nodal analysis (MNA) approach, in the following general form [18]:

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \tag{1}$$

where $\boldsymbol{x}$ is the vector of node-voltage and branch-current variables, $\boldsymbol{A}$ is the modified nodal admittance matrix or simply the *circuit matrix*, and $\boldsymbol{b}$ represents the external sources.

The circuit hierarchy can be viewed as a rooted tree shown in Fig. 1. A circuit may have one or more subcircuits at each hierarchical level. A subcircuit at a leaf in the circuit hierarchy tree is called a *leaf* subcircuit, otherwise it is a *middle* subcircuit. In this paper, we assume the presence of the predefined *subcircuits* in the circuit hierarchy.

Consider a subcircuit with some internal structures and terminals, as illustrated in Fig. 2. The circuit unknowns—the node-voltage variables and branch-current variables—can be partitioned into three disjoint groups $\boldsymbol{x}^I$, $\boldsymbol{x}^B$, and $\boldsymbol{x}^R$, where the superscripts I, B, R stand for, respectively, *internal* variables, *boundary* variables, and the *rest* of variables. *Internal* variables are those local to the subcircuit, *boundary* variables (also called *tearing variables*) are those related to both the subcircuit and the

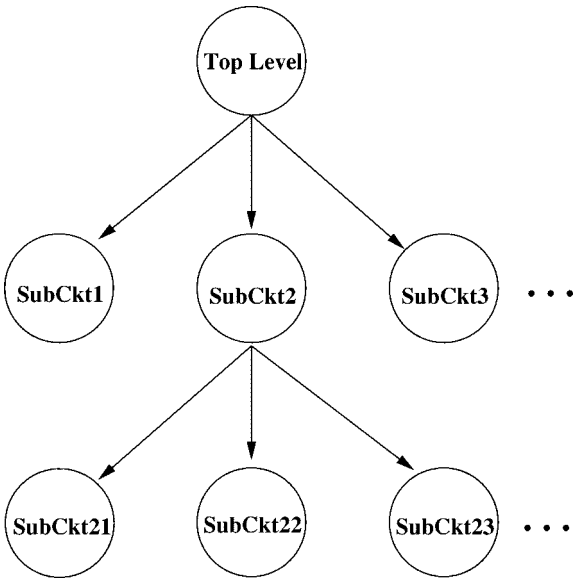


Fig. 1. Model of a circuit hierarchy.

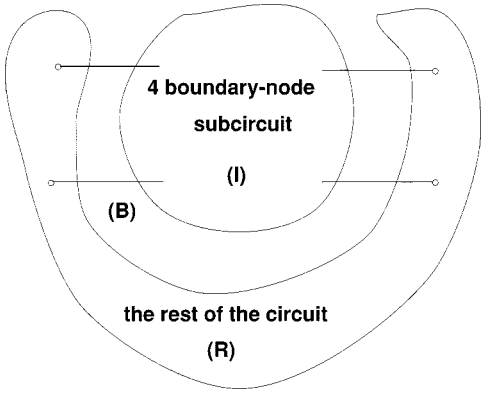


Fig. 2. Partition of a circuit.

rest of the circuit. Note that boundary variables include those variables required as the circuit inputs and outputs. Equations that are associated with only the *internal* variables are called the *internal* equations of a subcircuit. Their corresponding circuit matrix is called the *internal* circuit matrix. With this, the system-equation set (1) can be rewritten in the following form:

$$\begin{bmatrix} \mathbf{A}^{II} & \mathbf{A}^{IB} & 0 \\ \mathbf{A}^{BI} & \mathbf{A}^{BB} & \mathbf{A}^{BR} \\ 0 & \mathbf{A}^{RB} & \mathbf{A}^{RR} \end{bmatrix} \begin{bmatrix} \mathbf{x}^I \\ \mathbf{x}^B \\ \mathbf{x}^R \end{bmatrix} = \begin{bmatrix} \mathbf{b}^I \\ \mathbf{b}^B \\ \mathbf{b}^R \end{bmatrix}. \tag{2}$$

The gray matrix, $\boldsymbol{A}^{II}$, is the *internal* matrix associated with internal variable vector $\boldsymbol{x}^I$.

Subcircuit suppression is to eliminate all the variables in $\boldsymbol{x}^I$, and to transform (2) into the following reduced set of equations:

$$\begin{bmatrix} \boldsymbol{A}^{BB*} & \boldsymbol{A}^{BR} \\ \boldsymbol{A}^{RB} & \boldsymbol{A}^{RR} \end{bmatrix} \begin{bmatrix} \boldsymbol{x}^B \\ \boldsymbol{x}^R \end{bmatrix} = \begin{bmatrix} \boldsymbol{b}^{B*} \\ \boldsymbol{b}^R \end{bmatrix} \tag{3}$$

where

$$\boldsymbol{A}^{BB*} = \boldsymbol{A}^{BB} - \boldsymbol{A}^{BI}\left(\boldsymbol{A}^{II}\right)^{-1}\boldsymbol{A}^{IB} \tag{4}$$

and

$$\boldsymbol{b}^{B*} = \boldsymbol{b}^{B} - \boldsymbol{A}^{BI}\left(\boldsymbol{A}^{II}\right)^{-1}\boldsymbol{b}^{I}. \tag{5}$$

Subcircuit suppression can be performed for all the subcircuits by visiting the circuit hierarchy in a bottom-up fashion. Hierarchical *numerical* analysis performs (4) and (5) numerically by partial triangular decomposition [17]. Hierarchical *symbolic* analysis uses intermediate variables to represent (4) and (5), which leads to a *sequence of expressions* [6], [14]. In the network formulation approach [6], internal variables are suppressed one at a time. Hence, $\boldsymbol{A}^{II}$ becomes a scalar—a_{ii}, $\boldsymbol{A}^{BI}$ becomes a matrix with a single column, denoted as $\boldsymbol{A}^{Bi}$, and $\boldsymbol{A}^{IB}$ becomes a matrix with a single row, denoted as $\boldsymbol{A}^{iB}$. With this notation, (4) becomes

$$\boldsymbol{A}^{BB*} = \boldsymbol{A}^{BB} - \frac{1}{a_{ii}}\boldsymbol{A}^{Bi}\boldsymbol{A}^{iB}. \tag{6}$$

III. Hierarchical Analysis Using Determinants and Cofactors

In this section, we show how hierarchical symbolic circuit analysis can be represented using determinants and cofactors. We first introduce some notations. Let $\boldsymbol{A}$ be an $n \times n$ matrix. It may be denoted as $[a_{u,v}]$, $u, v = 1, \cdots, n$. Similarly, a vector $\boldsymbol{b}$ of size n is denoted as $[b_u]$, $u = 1, \cdots, n$. The *determinant* of matrix $\boldsymbol{A}$ is denoted by $det(\boldsymbol{A})$. According to linear algebra, the inverse of nonsingular matrix $\boldsymbol{A}$ can be written as

$$\boldsymbol{A}^{-1} = \frac{1}{\det(\boldsymbol{A})}\left[\Delta_{u,v}\right]^{T} \tag{7}$$

where matrix $[\Delta_{u,v}]^T$ is the transpose of matrix $[\Delta_{u,v}]$, and

$$\Delta_{u,v} = (-1)^{u+v}\det(\boldsymbol{A}_{a_{u,v}}). \tag{8}$$

Here, $[\Delta_{u,v}]^T$ is called the *adjoint* matrix of $\boldsymbol{A}$, $\Delta_{u,v}$ is the first-order *cofactor* of $\det(\boldsymbol{A})$ with respect to $a_{u,v}$, and matrix $\boldsymbol{A}_{a_{u,v}}$ is the $(n-1) \times (n-1)$-matrix obtained from matrix $\boldsymbol{A}$ by deleting row u and column v. Matrix $\boldsymbol{A}_{a_{u,v}}$ is sometimes written as $\boldsymbol{A}_{u,v}$ in the sequel. Note that each entry in the adjoint matrix is a first-order cofactor of the original matrix, and the adjoint matrix itself is a *full* (dense) matrix.

Now we consider subcircuit suppression. Applying (7) to (4) and (5), we have

$$\boldsymbol{A}^{BB*} = \boldsymbol{A}^{BB} - \frac{1}{\det\left(\boldsymbol{A}^{II}\right)}\boldsymbol{A}^{BI}\left[\Delta^{II}_{u,v}\right]^{T}\boldsymbol{A}^{IB} \tag{9}$$

and

$$\boldsymbol{b}^{B*} = \boldsymbol{b}^{B} - \frac{1}{\det\left(\boldsymbol{A}^{II}\right)}\boldsymbol{A}^{BI}\left[\Delta^{II}_{u,v}\right]^{T}\boldsymbol{b}^{I}. \tag{10}$$

Suppose that the number of internal variables is m, and the number of boundary variables is l. Equations (9) and (10) can be written in the following expanded forms:

$$a^{BB*}_{u,v} = a^{BB}_{u,v} - \frac{1}{\det\left(\boldsymbol{A}^{II}\right)}\sum_{k_1,k_2=1}^{m} a^{BI}_{u,k_1}\Delta^{II}_{k_2,k_1}a^{IB}_{k_2,v}, \quad u, v = 1, \cdots, l \tag{11}$$

and

$$b^{B*}_{u} = b^{B}_{u} - \frac{1}{\det\left(\boldsymbol{A}^{II}\right)}\sum_{k_1,k_2=1}^{m} a^{BI}_{u,k_1}\Delta^{II}_{k_2,k_1}b^{I}_{k_2}, \quad u, = 1, \cdots, l. \tag{12}$$

From (11) and (12), we can observe that first-order cofactors $\Delta^{II}_{k_2,k_1}$ are required only when both a_{u,k_1} and $a_{k_2,v}$ are nonzeros. For practical circuits, l usually is much smaller than m provided that a good circuit partition is given, and $\boldsymbol{A}^{BI}$ and $\boldsymbol{A}^{IB}$ are generally very *sparse*. This implies that only *a very few* of the first-order cofactors of $\boldsymbol{A}^{II}$ are needed for subcircuit suppression.

At the top level of the circuit hierarchy tree, we can simply use Cramer's rule to obtain the desired transfer function. For example, suppose that the reduced equation set for the root circuit is $\boldsymbol{A}'\boldsymbol{x}' = \boldsymbol{b}'$, then the voltage gain from node i to node t can be expressed as follows:

$$H(s)_{it} = \frac{V_t(s)}{V_i(s)} = \frac{(-1)^{i+t}\det\left(\boldsymbol{A}'_{it}\right)}{(-1)^{i+i}\det\left(\boldsymbol{A}'_{ii}\right)} \tag{13}$$

where $\boldsymbol{A}'_{uv}$ is the submatrix obtained from $\boldsymbol{A}'$ by deleting row u and column v.

The key idea of the proposed method for hierarchical symbolic analysis is to represent all the determinants and cofactors in (11)–(13) by a newly introduced graph, called DDD's. DDD's exploit systematically the expression sharing among the determinants of subcircuit matrices and the required first-order cofactors. The exploration thereby leads to a very compact representation of transfer functions and renders DDD's extremely suitable for hierarchical symbolic analysis.

IV. Determinant Decision Diagrams

A DDD is a canonical and compact graphical representation of a symbolic-matrix determinant [11], [12]. It is a signed, rooted, directed, and acyclic graph. Each DDD vertex represents the determinant of a symbolic matrix. It has two outgoing edges pointing to two children vertices. Similar to binary decision diagrams (BDD's) for Shannon expansion of Boolean functions, DDD is a graphical representation of the following expansion of a matrix determinant:

$$\det(\boldsymbol{A}) = a_{r,c}(-1)^{r+c}\det\left(\boldsymbol{A}_{a_{r,c}}\right) + \det\left(\boldsymbol{A}_{\overline{a}_{r,c}}\right) \tag{14}$$

where

$a_{r,c}$	is the matrix element at row r;
column c of matrix $\boldsymbol{A}$,	
$(-1)^{r+c}\det(\boldsymbol{A}_{a_{r,c}})$	is the first-order cofactor of $\det(\boldsymbol{A})$ with respect to $a_{r,c}$;
$\det(\boldsymbol{A}_{\overline{a}_{r,c}})$	is the *remainder* of $\det(\boldsymbol{A})$ with respect to $a_{r,c}$.

Matrix $\boldsymbol{A}_{a_{r,c}}$ can be obtained from matrix $\boldsymbol{A}$ by deleting row r and column c. Matrix $\boldsymbol{A}_{\overline{a}_{r,c}}$ can be obtained by setting $a_{r,c}$ to zero in $\boldsymbol{A}$. Note that $\det(\boldsymbol{A}_{a_{r,c}})$ is also called a *minor* of $\det(\boldsymbol{A})$ with respect to $a_{r,c}$.

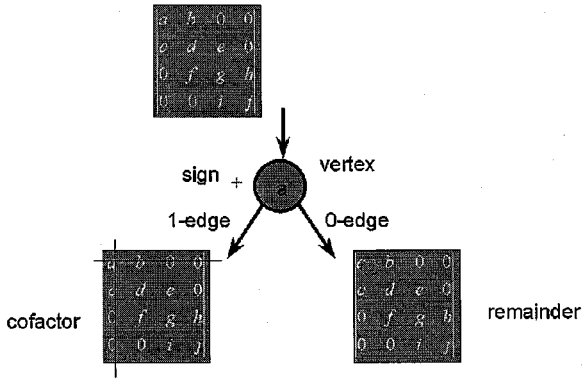


Fig. 3. Graphical representation of a determinant expansion.

In the DDD representation, we label the vertex by $a_{r,c}$ and assign the vertex sign as $(-1)^{r+c}$. We use the two children vertices to represent minor $\det(\boldsymbol{A}_{a_{r,c}})$ and remainder $\det(\boldsymbol{A}_{\overline{a}_{r,c}})$. We use a *1-edge* to link the vertex representing $\det(\boldsymbol{A})$ to the vertex representing $\det(\boldsymbol{A}_{a_{r,c}})$ and a *0-edge* to link the vertex representing $\det(\boldsymbol{A})$ to the vertex representing $\det(\boldsymbol{A}_{\overline{a}_{r,c}})$. This is illustrated in Fig. 3. This expansion process can be recursively performed on $\det(\boldsymbol{A}_{a_{r,c}})$ and $\det(\boldsymbol{A}_{\overline{a}_{r,c}})$. This leads to a binary decision diagram with two terminal vertices, namely the zero-terminal vertex representing constant zero and the one-terminal vertex representing constant one.

For example, consider the following determinant

$$\det(\boldsymbol{M})=\begin{vmatrix} a & b & 0 & 0 \\ c & d & e & 0 \\ 0 & f & g & h \\ 0 & 0 & i & j \end{vmatrix} = adgj-adhi-aefj-bcgj+bchi. \tag{15}$$

Fig. 4 illustrates the corresponding DDD representation under the expansion order: a, c, b, d, f, e, g, i, h, and j. Symbolic expressions represented by each vertex are also given near the vertices in the figure.

In a DDD, each path from the root vertex (a in our case) to the 1-terminal is called a *1-path*. Each 1-path defines a product term which includes all vertices (symbols) which originate all the 1-edges in the 1-path. We note that in Fig. 4 subterms ad, gj, and hi appear in several product terms of the matrix determinant, and they are shared in the DDD representation.

A key issue is that how to find a suitable expansion order for a given circuit matrix so that the resulting DDD has as few vertices as possible. A simple and efficient heuristic is to first expand those matrix rows or columns with fewest numbers of nonzero entries. It has been proved that this simple heuristic yields optimal DDD's for a class of circuits in the sense that for each circuit, the number of DDD vertices is exactly equal to the number of nonzero elements in the circuit matrix [11], [12]. We emphasize that in the worst case, the number of DDD vertices can grow exponentially with the size of a circuit. However, for practical analog circuits, the numbers of DDD vertices are generally many orders of magnitude less than the numbers of product terms [11], [12].

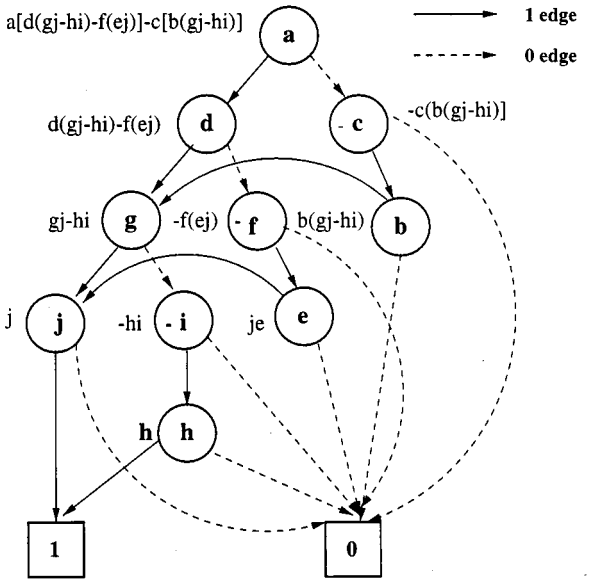


Fig. 4. A DDD for matrix $\boldsymbol{M}$.

We note that the first program that uses determinant expansion for symbolic circuit analysis is ISAAC [19]. ISAAC expands the determinant and minors recursively and uses a cache to avoid duplicate constructions of the same minor. Later on, the program SAGA developed by Jou and Hung improved ISAAC by combining top-down determinant expansion and bottom-up minor construction to reduce the number of symbolic multiplications [8]. Although DDD's are constructed based on a similar determinant expansion procedure as in ISAAC and SAGA, DDD's achieve the advantage by formulating the expansion process as a graph, and using the graph to represent symbolic expressions. With this, the problem of symbolic analysis reduces to the problem of graph manipulation, which has the time complexity proportional to the number of DDD vertices, not the number of product terms. The formalization of DDD's also allows a systematic exploration of expression sharing and matrix sparsity.

V. An Illustration Example

In this section, we illustrate the proposed hierarchical analysis method using a real analog circuit. We consider a Cauer parameter low-pass filter with 0.02-dB ripple in the passband and minimum 50-dB suppress in stopband as shown in Fig. 5. It has four topologically identical frequency-dependent negative resistance (FDNR) subcircuits, named $X1$–$X4$. Fig. 6 gives the detailed structure of the FDNR subcircuit. An FDNR subcircuit contains two operational amplifiers (opamp), which are implemented by a well-known linear macromodel shown in Fig. 7.

We first consider the linear macromodel of an opamp in Fig. 7. There are four nodes in this leaf subcircuit and, thus, four variables v_i, $i = 1, \ldots, 4$ in the circuit equations. Except v_1, all variables are boundary variables. Its circuit matrix $\boldsymbol{T}^o$,

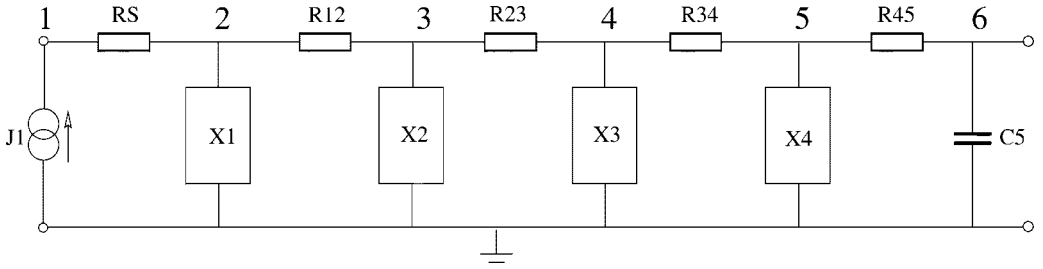


Fig. 5. An active low-pass filter.

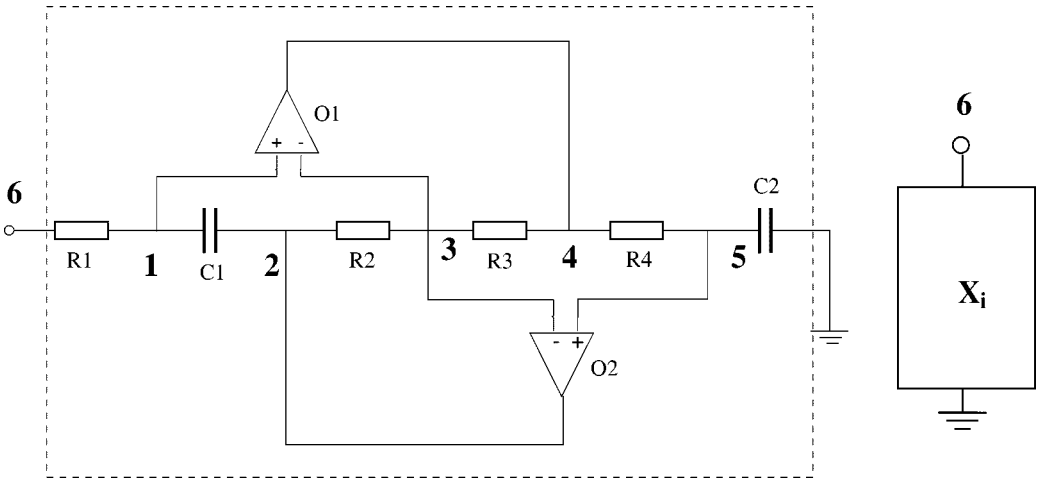


Fig. 6. An FDNR subcircuit.

under the modified nodal analysis formulation, can be written as follows:

$$\mathbf{T}^o = \begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \end{array} \overset{\overset{(\mathbf{T}^o)^{II}}{\Downarrow}}{\begin{bmatrix} sC_1+\frac{1}{R_1} & g_1 & -g_1 & 0 \\ 0 & \frac{1}{R_{ii}}+\frac{1}{R_{i1}} & -\frac{1}{R_{ii}} & 0 \\ 0 & -\frac{1}{R_{ii}} & \frac{1}{R_{ii}}+\frac{1}{R_{i2}} & 0 \\ g_2 & 0 & 0 & -\frac{1}{R_{id}} \end{bmatrix}}. \tag{16}$$

The gray variable is the internal variable to be suppressed. The gray matrix, $(\boldsymbol{T}^o)^{II}$, in the $\boldsymbol{T}^o$ is the internal circuit matrix of the opamp macromodel subcircuit, i.e., $(\boldsymbol{T}^o)^{II} = [sC_1 + (1/R_1)]$.

To suppress an opamp macromodel subcircuit, we need to suppress variable v_1. According to (11), the suppressed circuit matrix associated with the boundary variables is given by

$$(\mathbf{T}^o)^{BB*} = \begin{array}{c} \\ v_2 \\ v_3 \\ v_4 \end{array} \underbrace{\begin{bmatrix} \frac{1}{R_{ii}}+\frac{1}{R_{i1}} & -\frac{1}{R_{ii}} & 0 \\ -\frac{1}{R_{ii}} & \frac{1}{R_{ii}}+\frac{1}{R_{i2}} & 0 \\ (t^o_{42})^{BB*} & (t^o_{43})^{BB*} & -\frac{1}{R_{id}} \end{bmatrix}}_{\text{fill-ins}} \tag{17}$$

where

$$\begin{aligned} (t^o_{42})^{BB*} &= -\frac{1}{\det\left((\boldsymbol{T}^o)^{II}\right)} g_2 \left(\Delta^o_{11}\right)^{II} g_1 \\ &= -\frac{g_2 g_1}{sC_1 + \dfrac{1}{R_1}} \end{aligned} \tag{18}$$

$$\begin{aligned} (t^o_{43})^{BB*} &= -\frac{1}{\det\left((\boldsymbol{T}^o)^{II}\right)} g_2 \left(\Delta^o_{11}\right)^{II} (-g_1) \\ &= \frac{g_2 g_1}{sC_1 + \dfrac{1}{R_1}} \end{aligned} \tag{19}$$

and $(\Delta^o_{11})^{II}$ is the first-order cofactor of $(\boldsymbol{T}^o)^{II}$ with respect to the row 1 and column 1 of $(\boldsymbol{T}^o)^{II}$. Note that

$$\left(\Delta^o_{11}\right)^{II} = (-1)^{(1+1)} \det\left(\left(\boldsymbol{T}^o_{11}\right)^{II}\right) = 1 \tag{20}$$

where $(\boldsymbol{T}^o_{11})^{II} = 1$. Thus the suppression of an opamp macromodel subcircuit requires the DDD representation of two determinants: $\det((\boldsymbol{T}^o)^{II})$ and $\det((\boldsymbol{T}^o_{11})^{II})$. Since $\det((\boldsymbol{T}^o_{11})^{II})$ is constant 1, only one DDD vertex is needed; the resulting DDD is shown in Fig. 8 with $a = sC_1 + 1/R_1$.

After the suppression of all opamp macromodel subcircuits, we are ready to consider their parents, FDNR subcircuits. The circuit matrix of an FDNR subcircuit can be constructed by combining the matrix in (17) for an opamp macromodel subcircuit with the contributions from resistors in the FDNR circuit.

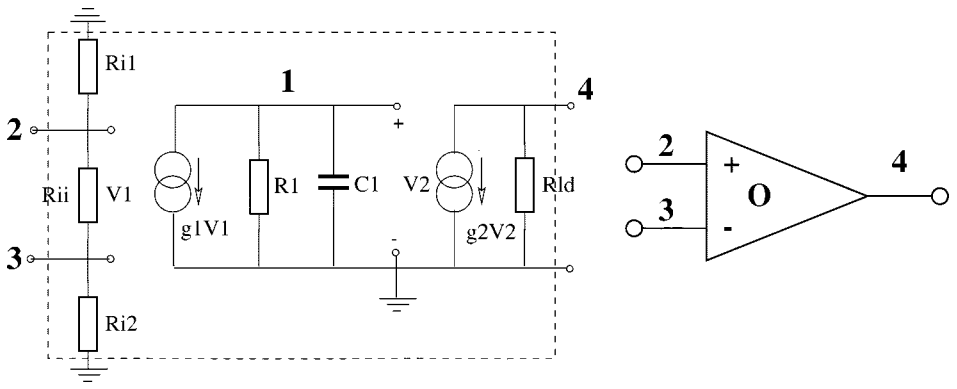


Fig. 7. A linear macromodel of an opamp.

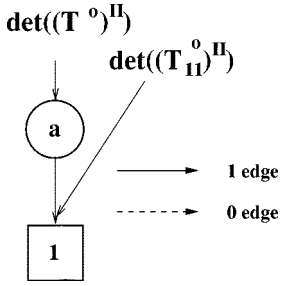


Fig. 8. The DDD for $\det((T^o)^{II})$ and $\det((T^o_{11})^{II})$.

The resulting circuit matrix can be written as (21), shown at the bottom of the page, where

$$\begin{aligned} p^{l2}_{11} &= \frac{1}{R_1} + sC_1 + \frac{1}{R_{ii}} + \frac{1}{R_{i1}} \\ p^{l2}_{22} &= sC_1 + \frac{1}{R_2} + \frac{1}{R_{ld}} \\ p^{l2}_{33} &= \frac{1}{R_2} + \frac{1}{R_3} + \frac{2}{R_{ii}} + \frac{2}{R_{i2}} \\ p^{l2}_{44} &= \frac{1}{R_3} + \frac{1}{R_4} + \frac{1}{R_{ld}} \\ p^{l2}_{55} &= \frac{1}{R_4} + sC_2 + \frac{1}{R_{ii}} + \frac{1}{R_{i1}} \\ p^{l2}_{66} &= \frac{1}{R_1}. \end{aligned} \tag{22}$$

Again, the gray variables are internal variables to be suppressed, and the gray matrix, $(\boldsymbol{T}^x)^{II}$, in $\boldsymbol{T}^x$ is the internal circuit matrix of an FDNR subcircuit. The suppressed circuit matrix of $\boldsymbol{T}^x$, which is associated with only the boundary variable v_6, becomes a 1×1 matrix as follows:

$$(\boldsymbol{T}^x)^{BB*} = \left[p^{l2}_{66} + (t^x_{66})^{BB*} \right] \tag{23}$$

where

$$(t^x_{66})^{BB*} = -\frac{1}{\det\left((\boldsymbol{T}^x)^{II}\right)} \left(-\frac{1}{R_1}\right) (\Delta^x_{11})^{II} \left(-\frac{1}{R_1}\right). \tag{24}$$

Here, $(\Delta^x_{11})^{II}$ is the first-order cofactor of $(\boldsymbol{T}^x)^{II}$ with respect to the element in row 1 and column 1 of $(\boldsymbol{T}^x)^{II}$ and it can be written as

$$\begin{aligned} (\Delta^x_{11})^{II} &= (-1)^{(1+1)} \det\left((\boldsymbol{T}^x_{11})^{II}\right) \\ &= \begin{vmatrix} p^{l2}_{22} & -\frac{1}{R_2} + (t^o_{43})^{BB*} & 0 & (t^o_{42})^{BB*} \\ -\frac{1}{R_2} & p^{l2}_{33} & -\frac{1}{R_3} & -\frac{1}{R_{ii}} \\ 0 & -\frac{1}{R_3} + (t^o_{43})^{BB*} & p^{l2}_{44} & -\frac{1}{R_4} \\ 0 & -\frac{1}{R_{ii}} & -\frac{1}{R_4} & p^{l2}_{55} \end{vmatrix}. \end{aligned}$$

$$\mathbf{T}^x = \begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \end{array} \overbrace{\begin{bmatrix} p^{l2}_{11} & -sC_1 & -\frac{1}{R_{ii}} & 0 & 0 & -\frac{1}{R_1} \\ -sC_1 & p^{l2}_{22} & -\frac{1}{R_2} + (t^o_{43})^{BB*} & 0 & (t^o_{42})^{BB*} & 0 \\ -\frac{1}{R_{ii}} & -\frac{1}{R_2} & p^{l2}_{33} & -\frac{1}{R_3} & -\frac{1}{R_{ii}} & 0 \\ (t^o_{42})^{BB*} & 0 & -\frac{1}{R_3} + (t^o_{43})^{BB*} & p^{l2}_{44} & -\frac{1}{R_4} & 0 \\ 0 & 0 & -\frac{1}{R_{ii}} & -\frac{1}{R_4} & p^{l2}_{55} & 0 \\ -\frac{1}{R_1} & 0 & 0 & 0 & 0 & p^{l2}_{66} \end{bmatrix}}^{(\mathbf{T}^x)^{II}} \tag{21}$$

We now show how the two required determinants, $\det((\boldsymbol{T}^x)^{II})$ and $\det((\boldsymbol{T}^x_{11})^{II})$, can be represented by DDD's. To simplify our presentation, we label each matrix entry in $(\boldsymbol{T}^x)^{II}$ with a distinct symbol as

$$det((\mathbf{T}^x)^{II}) = \begin{vmatrix} a & b & c & 0 & 0 \\ d & e & f & 0 & g \\ h & i & j & k & l \\ m & 0 & n & o & p \\ 0 & 0 & q & r & s \end{vmatrix}$$

where, the gray area is matrix $(\boldsymbol{T}^x_{11})^{II}$. Determinant $\det((\boldsymbol{T}^x)^{II})$ has 31 product terms, and $\det((\boldsymbol{T}^x_{11})^{II})$ has 10 product terms. Detailed analysis shows that they can be represented by two DDD's with 37 and 19 vertices, respectively. Exploiting the sharing of product terms among these two DDD's, the two determinants can actually be represented by a shared DDD with only 39 vertices as shown in Fig. 9. This result contrasts with 247 ($=37\times 5+19\times 4$) DDD vertices if each symbol is represented by a DDD vertex without sharing. Finally, the circuit matrix of the root circuit, the low-pass filter, is constructed after the suppression of all FDNR subcircuits. The resulting circuit matrix is given

$$\boldsymbol{T} = \begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \end{array} \begin{bmatrix} p^{l1}_{11} & -\frac{1}{R_s} & 0 & 0 & 0 & 0 \\ -\frac{1}{R_s} & p^{l1}_{22} & -\frac{1}{R_{12}} & 0 & 0 & 0 \\ 0 & -\frac{1}{R_{12}} & p^{l1}_{33} & -\frac{1}{R_{23}} & 0 & 0 \\ 0 & 0 & -\frac{1}{R_{23}} & p^{l1}_{44} & -\frac{1}{R_{34}} & 0 \\ 0 & 0 & 0 & -\frac{1}{R_{34}} & p^{l1}_{55} & -\frac{1}{R_{45}} \\ 0 & 0 & 0 & 0 & -\frac{1}{R_{45}} & p^{l1}_{66} \end{bmatrix} \tag{25}$$

where

$$\begin{aligned} p^{l1}_{11} &= \frac{1}{R_s}, \\ p^{l1}_{22} &= \frac{1}{R_s} + \frac{1}{R_{12}} + \frac{1}{R_1} + (t^x_{66})^{BB*}_1 \\ p^{l1}_{33} &= \frac{1}{R_{12}} + \frac{1}{R_{23}} + \frac{1}{R_1} + (t^x_{66})^{BB*}_2 \\ p^{l1}_{44} &= \frac{1}{R_{23}} + \frac{1}{R_{34}} + \frac{1}{R_1} + (t^x_{66})^{BB*}_3 \\ p^{l1}_{55} &= \frac{1}{R_{34}} + \frac{1}{R_{45}} + \frac{1}{R_1} + (t^x_{66})^{BB*}_4 \\ p^{l1}_{66} &= \frac{1}{R_{45}} + sC_5. \end{aligned}$$

Note that each FDNR subcircuit may have a different set of device parameter values. Thus $(t^x_{66})^{BB*}_i$, $i=1,\ldots,4$, are used to represent $(t^x_{66})^{BB*}$ for four FDNR subcircuits. Since all the FDNR subcircuits have the same topology, only one symbolic expression of $(t^x_{66})^{BB*}$ and, thus, one DDD (Fig. 8), is needed.

The required transfer function can now be derived and represented by DDD's. According to Cramer's rule, the voltage gain from V_1–V_6 in Fig. 5 can be expressed as

$$G(s)_{16} = \frac{V_6}{V_1} = \frac{(-1)^{1+6}\det(\boldsymbol{T}_{16})}{(-1)^{1+1}\det(\boldsymbol{T}_{11})}.$$

To illustrate the DDD representation of the transfer function, we label each matrix entry in $\boldsymbol{T}$ with a distinct symbol and rewrite $\boldsymbol{T}$ as follows:

$$det(\mathbf{T}) = \begin{vmatrix} a & b & 0 & 0 & 0 & 0 \\ c & d & e & 0 & 0 & 0 \\ 0 & f & g & h & 0 & 0 \\ 0 & 0 & i & j & k & 0 \\ 0 & 0 & 0 & l & m & n \\ 0 & 0 & 0 & 0 & o & p \end{vmatrix}$$

where the boxed matrix is $\boldsymbol{T}_{16}$ and the gray matrix is $\boldsymbol{T}_{11}$. Note that $\boldsymbol{T}_{11}$ is a 5×5 band matrix. The DDD representation of $\det(\boldsymbol{T}_{11})$ and $\det(\boldsymbol{T}_{16})$ is shown in Fig. 10, where 13 DDD vertices are used to represent $\det(\boldsymbol{T}_{11})$, and 5 DDD vertices (each for a matrix entry) are needed to represent $\det(\boldsymbol{T}_{16})$. Taking into account of the DDD's for subcircuit suppression (Figs. 7 and 8), a total of 58 ($=1+29+18$) DDD vertices are used for entire hierarchical symbolic analysis of the low-pass filter circuit.

To conclude this section, we have the following observations.

- Suppression of a subcircuit may create fill-ins in its parent circuit matrix. For instance, $(t^o_{42})^{BB*}$ appearing in row 2 and column 5, as well as in row 4 and column 1, in matrix $(\boldsymbol{T}^x)^{II}$ in (21) are fill-ins. In order to obtain a compact symbolic expression, the number of fill-ins should be minimized. This is generally consistent with minimizing the total number of *boundary* nodes of subcircuits. An efficient heuristic based on this idea has been developed for finding a good partition for DDD-based hierarchical symbolic analysis [16]. We note that the problem of automatic recognition of identical subcircuits remains open.
- Given a good partition of an analog circuit, subcircuit suppression only requires a few first-order cofactors of the subcircuit-matrix determinants. In our example, only one cofactor, $(t^x_{66})^{BB*}$, is actually required in the entire analysis process.

VI. Hierarchical Symbolic Analysis Procedure

The proposed hierarchical symbolic analysis method is performed by the depth-first traversal of the circuit hierarchy tree shown in Fig. 1. Then DDD's are constructed for each subcircuit

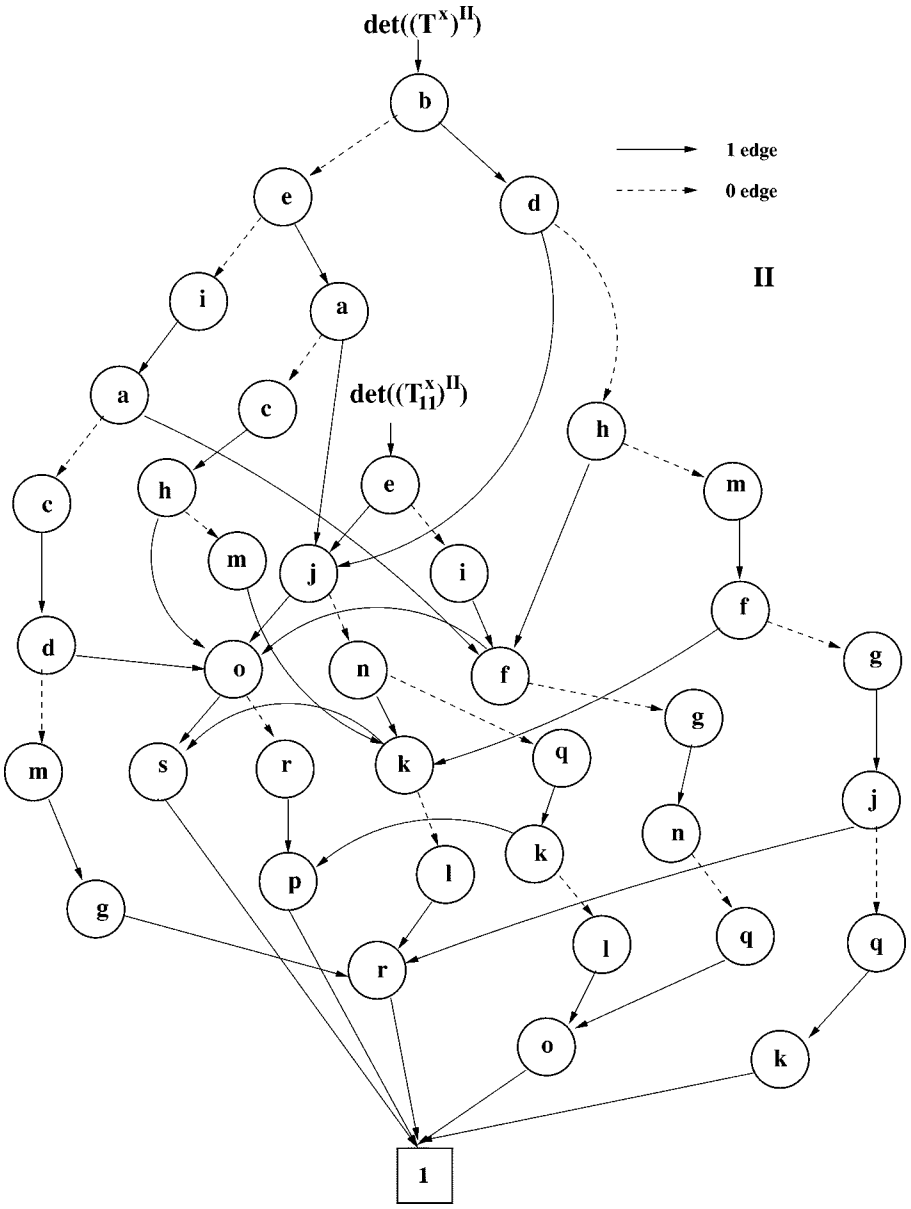


Fig. 9. The DDD for $\det((\boldsymbol{T}^x)^{II})$ and $\det((\boldsymbol{T}^x_{11})^{II})$.

based on the algorithms described in [11] and [12]. The complete procedure for DDD-based hierarchical symbolic analysis can be summarized as follows.

1) Partition the circuit or use the predefined subcircuit structure.
2) Build the circuit matrix for each leaf subcircuit by modified nodal analysis. Suppress each leaf subcircuit based on (11) and (12). Represent the determinant of the internal subcircuit matrix and its first-order cofactors required in subcircuit suppression by a shared DDD.
3) Build the circuit matrix of a middle subcircuit after the suppression of all its children subcircuits. An entry in the middle subcircuit matrix may consist of the contribution from its children subcircuits as well as that from the circuit devices in the middle subcircuit. Suppress the middle subcircuit based on (11) and (12). Represent the determinant of the internal subcircuit matrix of the middle subcircuit and its first-order cofactors required in subcircuit suppression by a shared DDD. Recursively build and suppress the circuit matrices for all the middle subcircuits until the root circuit is reached.
4) Construct the desired symbolic transfer function at the root circuit by using Cramer's rule with all the required symbolic determinants and cofactors represented by DDD's.

VII. Experimental Results

The proposed method has been implemented in a symbolic circuit analyzer based on DDD's. The program reads in a circuit description in the SPICE format, where *.subckt* statements are used to specify the circuit hierarchy. All the MOS and bipolar transistors are replaced by their corresponding small-signal models at their dc operating points computed by SPICE. The ac

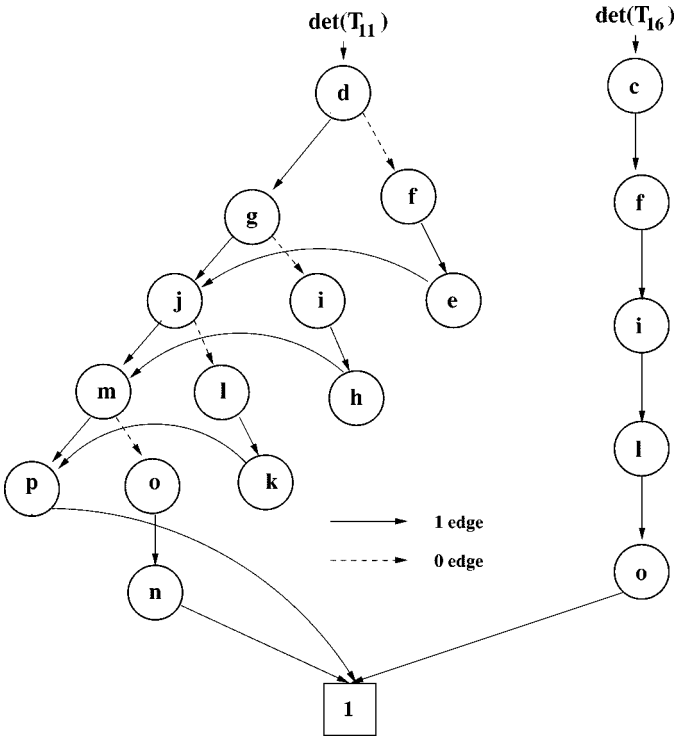


Fig. 10. The DDD for $\det(\boldsymbol{T}_{11})$ and $\det(\boldsymbol{T}_{16})$.

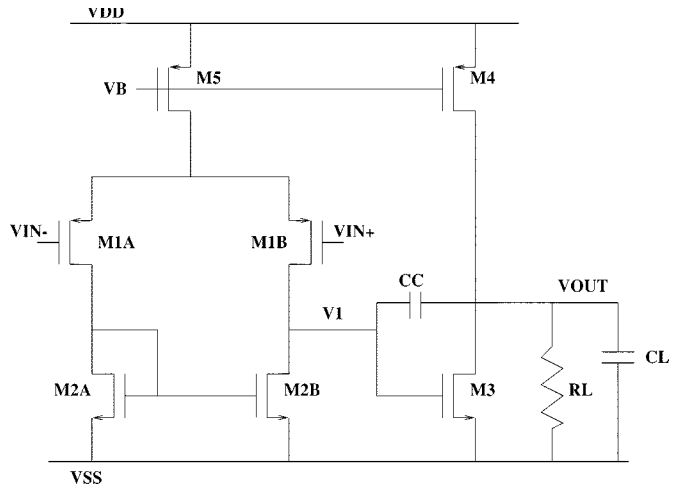


Fig. 11. Miller-compensated two-stage opamp.

analysis is performed by depth-first traversals of all the DDD vertices used to represent all symbolic expressions at each frequency point.

A number of experiments have been conducted on a SUN SPARCstation 5 with 32M memory. The results from three examples are presented. The first example is an active low-pass filter circuit shown in Fig. 5. We tested our program on two different implementations of opamps used in the low-pass filter circuit: the macromodel shown in Fig. 7 and a simplified miller-compensated two-stage opamp circuit shown in Fig. 11.

The second example is a bandpass filter circuit shown in Fig. 12. This example was also used to illustrate hierarchical analysis in [6] and [14]. The circuit can be partitioned into the circuit hierarchy shown in Fig. 13 with four topologically identical subcircuits $X1$–$X4$ shown in Fig. 14. Each leaf-level

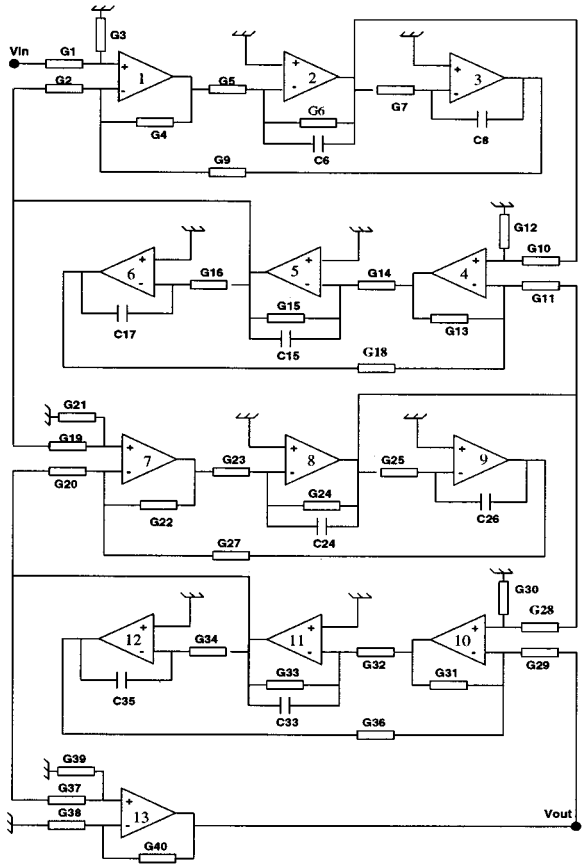


Fig. 12. An active RC bandpass filter.

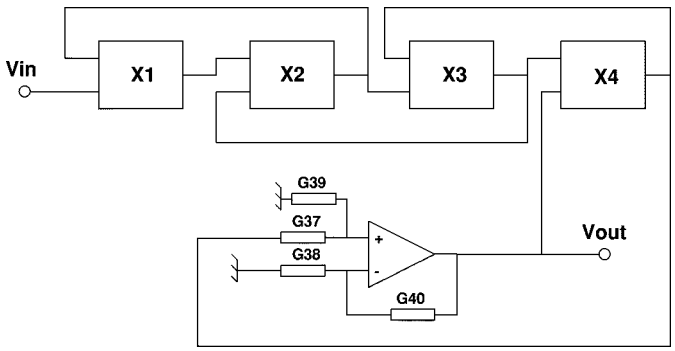


Fig. 13. The partitioned hierarchy of the bandpass filter.

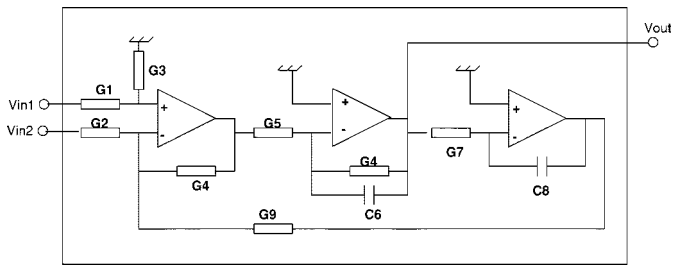


Fig. 14. The subcircuit structure in the bandpass filter.

opamp subcircuit is implemented by the miller-compensated two-stage opamp circuit shown in Fig. 11.

The third example is a μA741 opamp circuit with 26 transistors and 11 transistors shown in Fig. 15. Unlike previous

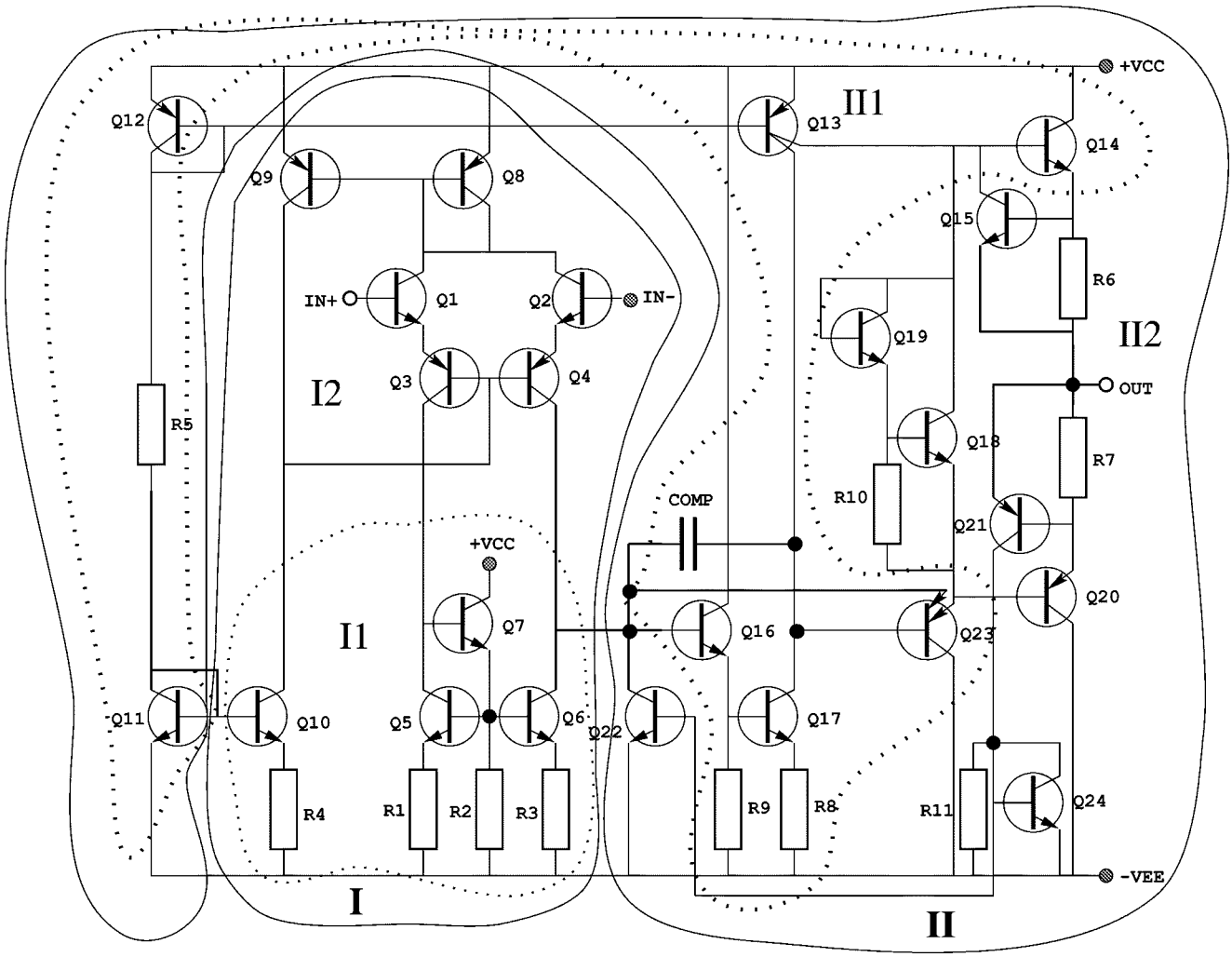


Fig. 15. A three-level two-way partitioned μA741.

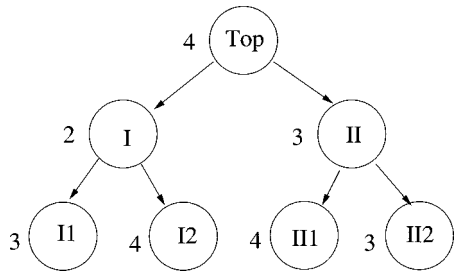


Fig. 16. A three-level two-way partition of μA741.

two examples, this circuit has less structural regularities. In this paper, we consider a three-level binary-tree hierarchy (Fig. 16) as marked in Fig. 15; this hierarchy is obtained by using a multilevel multiway partitioning heuristic aimed at minimizing the total number of DDD vertices [16].

We first compare our program to SPICE on repetitive numerical evaluation. For each circuit, 1000 frequency points are computed. The results are summarized in Table I, where columns 2 and 3 show, respectively, the size of the overall circuit matrix and the total number of nonzeros for each circuit, column 4 is the actual number of distinct product terms generated, column 5 is the total number of DDD vertices used to represent all the symbolic expressions. Columns 6–8 list, respectively, the simulation CPU time in seconds used by the proposed DDD-based method, SPICE, and the speedup of the proposed method over SPICE for each test circuit. From Table I, we can see that the proposed DDD-based method outperforms SPICE for all the test cases. Further, the speedup increases with the size of a circuit.

We then compare our method to SCAPP—a best-known hierarchical symbolic analyzer. We construct the test circuits by cascading, respectively, the first 1–4 subcircuit blocks (Fig. 13). The opamp subcircuit is implemented by the miller-compensated opamp circuit shown in Fig. 11.

The results are summarized in Table II. Columns 1 and 2 list, respectively, the number of subcircuits cascaded for each test case, and the size of the overall circuit matrix, and the total number of nonzeros. Columns 4 and 5 describe, respectively, the total number of DDD vertices and the number of product terms represented. Columns 6 and 7 give the total numbers of additions and multiplications used in the expressions generated by SCAPP. Since each DDD vertex uses one addition and one multiplication, the number of additions and multiplications used by the DDD-based method is bounded by the number of DDD vertices. From Table II, we can observe that the DDD-based representation is

TABLE I
COMPARISON AGAINST SPICE IN NUMERICAL EVALUATION

circuit	overall matrix size	#nonzeros	#term	|DDD|	DDD-based	SPICE	speedup
LP(linear)	31	121	52	58	1.07	6.58	6.15
LP(miller)	32	162	124	110	1.56	14.82	9.50
BP(linear)	38	147	456	173	3.07	9.70	3.16
BP(miller)	44	206	649	251	3.72	17.76	4.77

LP — low-pass filter; BP — band-pass filter.
linear or *miller* denotes the corresponding Opamp model used.

TABLE II
COMPARISON AGAINST SCAPP

			DDD-based		SCAPP	
#subckt	overall matrix size	#nonzeros	$\|DDD\|$	#terms	#add	#mul
1	23	89	142	500	556	274
2	39	154	367	2.60×10^4	1339	854
3	55	219	566	1.60×10^6	1772	1074
4	71	283	765	1.01×10^8	2479	1639

TABLE III
COMPARISON AGAINST SCAPP AND SPICE IN CPU TIME

	DDD-based		SCAPP			SPICE	
#subckt	const.	sim.	const.	comp.	sim.	setup	sim.
1	0.33	2.09	0.81	13.1	2.60	1.10	5.34
2	0.71	4.75	2.09	33.3	7.49	2.70	8.98
3	1.25	6.91	3.69	44.2	10.37	3.12	15.58
4	2.21	9.19	5.54	64.7	12.06	3.42	22.10

TABLE IV
STATISTICS FOR THREE-LEVEL HIERARCHICAL SYMBOLIC ANALYSIS OF μA741

w/o partitioning		overall matrix size	24			
		#nonzero	89			
		$\|DDD\|$	6654			
		#terms	119011			
w partitioning	leaf subcircuits		I1	I2	II1	II2
		matrix size	3	4	4	3
		$\|DDD\|$	10	36	23	6
	middle subcircuits		I		II	
		matrix size	2		3	
		#cut nets	5		6	
		$\|DDD\|$	4		18	
	root	$\|DDD\|$	20			
	total	$\|DDD\|$	117			
		#terms	219			
SCAPP		#mul:182, #add: 357				

much more compact than the sequence-of-expression representation used in SCAPP. In addition, we see that the DDD size grows almost linearly in the circuit size, although the number of product terms grow exponentially.

Table III shows the statistics of using the DDD-based symbolic method and SCAPP for repetitive numerical evaluation. In the current implementation of SCAPP, the sequence of expressions for circuit transfer functions are first generated as C code, the generated C code is then compiled, and the compiled code is finally linked with the simulation driver to perform ac analysis. For each test case, we report in columns 2 and 3 the CPU time required to construct the DDD and then the CPU time taken for simulating the frequency-domain response for 1000 frequency points from the constructed DDD. For SCAPP, we report, respectively, in columns 4, 5, and 6, the CPU time for SCAPP to construct the sequence-of-expressions (*const.*), the compilation time (*comp*), and the actual simulation time (*sim*). The last two columns give the matrix-setup time and simulation time used by SPICE.

From Table III, we can see that the proposed DDD-based method is more efficient than both SCAPP and SPICE. Note that, in our current implementation, the constructed DDD is stored in memory; hence, no additional compilation time is required. We note that SCAPP can be re-implemented to store the constructed sequences of expressions in memory, and then the extra compilation time can be avoided.

We finally test our program on a three-level two-way partitioned bipolar μA741 opamp circuit shown in Figs. 15 and 16. Table IV shows the statistics of hierarchical symbolic analysis of this circuit. The first four rows list the results of DDD-based symbolic analysis without partitioning, which include the size of the overall circuit matrix, the total number of nonzeros, the total number of DDD vertices, and the total product terms represented by the DDD. The next ten rows describe the statistics of three-level hierarchical symbolic analysis with partitioning; these results are broken down for leaf subcircuits, middle subcircuits, and the root circuit. The total number of DDD vertices and the number of terms generated are reported in rows 13 and 14. The last row shows the best result from SCAPP, where $\#mul$ and $\#add$ are the numbers of multiplications and additions, respectively. Since the number of multiplications required in numerically evaluating a DDD-based symbolic expression is bounded by the number of DDD vertices, three-level two-way DDD-based hierarchical symbolic analysis speeds up DDD-based canonical symbolic analysis (one-level one-way) by a factor of 56 (117 vertices versus 6654 vertices), whereas DDD-based canonical symbolic analysis already speeds up sum-of-product-based canonical symbolic analysis by several orders of magnitude (6654 multiplications versus 119 011 product term evaluation).

VIII. Conclusion

A new method for hierarchical symbolic analysis of analog integrated circuits has been presented and implemented. The method takes advantage of both the circuit hierarchy and DDD's for symbolic determinant representations. DDD representation exploits systematically the sharing among symbolic expressions and thus results in very compact symbolic expressions. Experimental results have shown that the proposed method compares favorably to the best-known symbolic analyzer SCAPP and numerical simulator SPICE for small-signal ac analysis.

Acknowledgment

The authors would like to thank Prof. G. Gielen of Katholieke Universiteit Leuven for several helpful discussions on symbolic analysis and Prof. M. Hassoun of Iowa State University for making SCAPP code available to them.

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*, 1992, pp. 1153–1156.

[2] F. V. Fernández, J. D. Martín, A. Rodríguez-Vázquez, and J. L. Huertas, "On simplification techniques for symbolic analysis of analog integrated circuits," in *Proc. IEEE Int. Symp. Circuits and Systems*, 1992, pp. 1149–1152.

[3] F. V. Fernández and A. Rodríguez-Vázquez, "Symbolic analysis tools—The state of the art," in *Proc. IEEE Int. Symp. Circuits and Systems*, 1996, pp. 798–801.

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

[5] G. Gielen, P. Wambacq, and W. Sansen, "Symbolic analysis methods and applications for analog circuits: A tutorial overview," *Proc. IEEE*, vol. 82, pp. 287–304, Feb. 1994.

[6] M. M. Hassoun and P. M. Lin, "A hierarchical network approach to symbolic analysis of large scale networks," *IEEE Trans. Circuits Syst.*, vol. 42, pp. 201–211, Apr. 1995.

[7] J.-J. Hsu and C. Sechen, "DC small signal symbolic analysis of large analog integrated circuits," *IEEE Trans. Circuits Syst.*, vol. 41, pp. 817–828, Dec. 1994.

[8] S.-J. Jou and C.-C. Hung, "Hierarchical symbolic analysis of analog circuits," in *Proc. National Science Council (R.O.C.)*, Part A, vol. 17, July 1993, pp. 301–313.

[9] A. Liberatore and S. Manetti, "SAPEC—A personnel computer program for the symbolic analysis of electric circuits," in *Proc. IEEE Int. Symp. Circuits and Systems*, 1988, pp. 897–900.

[10] P. M. Lin, "Sensitivity analysis of large linear networks using symbolic program," in *Proc. IEEE Int. Symp. Circuits and Systems*, 1992, pp. 1145–1148.

[11] C.-J. Shi and X.-D. Tan, "Symbolic analysis of large analog circuits with determinant decision diagrams," in *Proc. IEEE Int. Conf. Computer-Aided Design (ICCAD)*, 1997, pp. 366–373.

[12] ——, "Canonical symbolic analysis of large analog circuits with determinant decision diagrams," *IEEE Trans. Computer-Aided Design*, vol. 19, pp. 1–18, Jan. 2000.

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

[14] J. A. Starzky and A. Konczykowska, "Flowgraph analysis of large electronic networks," *IEEE Trans. Circuits Syst.*, vol. 33, pp. 302–315, Mar. 1986.

[15] X.-D. Tan and C.-J. Shi, "Hierarchical symbolic analysis of large analog circuits with determinant decision diagrams," in *Proc. IEEE Int. Symp. Circuits and Systems*, vol. VI, May 1998, pp. 318–321.

[16] ——, "Balanced multilevel multiway partitioning of large analog circuits for hierarchical symbolic analysis," in *Proc. IEEE Asia and South Pacific Design Automation Conf. (ASP-DAC)*, Hong Kong, 1999, pp. 1–4.

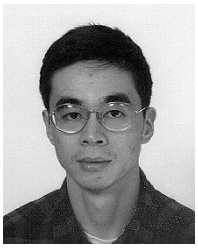
[17] M. Vlach, "LU decomposition algorithms for parallel and vector computation," in *Analog Methods for Computer-Aided Circuit Analysis and Diagnosis*, T. Ozawa, Ed. New York: Marcel Dekker, 1988, pp. 37–64.

[18] J. Vlach and K. Singhal, *Computer Methods for Circuit Analysis and Design*. New York: Van Nostrand Reinhold, 1994.

[19] H. Walscharts, G. Gielen, and W. Sansen, "Symbolic simulation of analog circuits in s- and z-domain," in *Proc. IEEE Int. Symp. Circuits and Systems*, 1989, pp. 814–817.

[20] 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*, 1996, pp. 584–587.

[21] Q. Yu and C. Sechen, "A unified approach to the approximate symbolic analysis of large analog integrated circuits," *IEEE Trans. Circuits Syst.*, vol. 43, pp. 656–669, Aug. 1996.



Xiang-Dong Tan (S'96–M'99) received the B.S. and M.S. degrees in electrical engineering from Fudan University, Shanghai, China, in 1992 and 1995, respectively, and the Ph.D. degree in electrical and computer engineering from the University of Iowa, Iowa City, in 1999.

He is currently a Member of Technical Staff at Monterey Design Systems, Monterey, CA. He worked with Rockwell Semiconductor Systems in the summer of 1997, and Avant! Corporation in the summer of 1998. He was a Research Assistant in the Department of Electrical Engineering, University of Washington, Seattle, from September 1998 to April 1999. His current research interests include very large scale integration (VLSI) physical design automation, symbolic analysis of large analog circuits, layout optimization for performance, timing, power, and clock tree synthesis.

Dr. Tan received a Best Paper Award from the 1999 IEEE/ACM Design Automation Conference in 1999 and the First-Place Student Poster Award from the 1999 Spring Meeting of the Center for Design of Analog Digital Integrated Circuits (CDADIC). He received a Best Graduate Award in 1992 and a number of Excellent College Student Scholarships from 1988 to 1992, all from Fudan University.

C.-J. Richard Shi (M'91–SM'99) received the B.S. and M.S. degrees in electrical engineering from Fudan University, Shanghai, China, in 1985 and 1988, respectively, the M.A.Sc. degree in electrical engineering and the Ph.D. degree in computer science from the University of Waterloo, Waterloo, ON, Canada, in 1991 and 1994, respectively.

He is currently an Assistant Professor in the Department of Electrical Engineering, University of Washington, Seattle. His research interests include methodologies and tools for systems-on-a-chip design, with the particular emphasis on analog, mixed-signal, and deep-submicron design and test automation. He has published more than 70 technical papers, and has been a principal investigator of more than $2M in research funding from DARPA, NSF, USAF, CDADIC, and industry since 1995. He is a consultant to several semiconductor and EDA companies.

Dr. Shi co-founded IEEE/ACM/VIUF International Workshop on Behavioral Modeling and Simulation, and served as its Technical Program Chair from 1997 to 1999. Having been involved in IEEE DASC 1076.1 VHDL-AMS Working Group since 1994, he is one of the contributors to, and promoters of, IEEE std 1076.1-1999 standard language (VHDL-AMS) for the description and simulation of mixed-signal/mixed-technology systems. He has delivered tutorials on VHDL-AMS and behavioral modeling at several conferences including DAC, EuroDAC, and ASP-DAC. He has been a recipient or co-recipient of several awards including the T. D. Lee Physics Award for excellence in graduate study from Fudan, University of Waterloo Outstanding Achievement in Graduate Studies Award, the Natural Sciences and Engineering Research Council of Canada Doctoral Prize, a National Science Foundation CAREER Award, four Best Paper Awards (including the 1999 IEEE/ACM Design Automation Conference Best Paper Award and the 1999 IEEE VLSI Test Symposium Best Paper Award), and three other Best Paper Award Nominations (ASP-DAC'98, EuroDAC'96, and ASP-DAC'95). He is a member of IEEE Design Automation Standards Committee. He is an Associate Editor of IEEE Transactions on Circuits and Systems–II. This year, he was nominated by his students to receive the UW/COE Outstanding Educator Award.