A Hierarchical Network Approach to Symbolic Analysis of Large-Scale Networks

Marwan M. Hassoun, *Member, IEEE,* and Pen-Min Lin, *Fellow, IEEE*

Abstract— **A hierarchical approach to the problem of symbolic circuit analysis of large-scale circuits is presented in this paper. The methodology has been implemented in a computer program called SCAPP (Symbolic Circuit Analysis Program with Partitioning). The method solves the problem by utilizing a hierarchical network approach and the *sequence of expressions* concept rather than a topological approach and the single expression idea which have dominated symbolic analysis in the past. The result is a linear growth (for real circuits) in the number of terms in the symbolic solutions for the network approach versus the exponential growth exhibited by traditional methods. The analysis methodology uses a Reduced Modified Nodal Analysis (RMNA) technique that allows the characterization of symbolic networks in terms of only a small subset of the network variables (external variables). The analysis algorithm is most efficient when network partitioning is used. Partitioning results in a reduction in the number of terms in the symbolic solutions.**

I. Introduction

THE IDEA of symbolic circuit analysis is that some, or all, of the circuit parameters remain as symbols (no numerical values assigned to them) throughout the entire simulation process. From a circuit design perspective, numerical results from the simulation of a circuit can be obtained by evaluating the results of the symbolic analysis at a specific numerical point for each symbol. So ideally, only one simulation run is needed in order to analyze the circuit, and successive evaluations of the results replace the need for any extra iterations through the simulator. Other applications include sensitivity analysis, circuit stability analysis, device modeling and circuit optimization [1]–[3].

Traditional symbolic circuit analysis is performed in the frequency domain where the results are in terms of the frequency variable s. The main goal of performing symbolic analysis on a network is to obtain a symbolic transfer function of the form

$$H(s, \boldsymbol{X}) = \frac{N(s, \boldsymbol{X})}{D(s, \boldsymbol{X})}, \quad \boldsymbol{X} = [x_1, x_2, \cdots, x_n], \qquad n \leq n_{\text{all}}. \tag{1}$$

Manuscript received April 8, 1993. This work was supported by the National Science Foundation Grant ECS-84-19841. This paper was recommended by Associate Editor D. Mlynski.

M. M. Hassoun is with the Department of Electrical and Computer Engineering, Iowa State University, Ames, IA 50011 USA.

P.-M. Lin is with the School of Electrical Engineering, Purdue University, West Lafayette, IN 47907 USA.

IEEE Log Number 9410011.

The expression is a function of the complex frequency variable s, and the variables x_1 through x_n representing the variable network parameters, where n is the number of variable network parameters and n_{all} is the total number of network parameters.

1.1. Symbolic Analysis Background

Several methods have been proposed to address the problem of symbolic circuit simulation [4]. The early work was to produce a transfer function $H(s)$ with the frequency variable s being the only symbolic variable. The more general case is when some or all of the circuit elements are represented by symbolic variables. The methods developed fall under one of the following categories [4]:

1) The tree enumeration method: Several programs have been produced based on this method [5], [6]. The process is based on the concept of finding the determinant of the node admittance matrix [7] by finding the sum of all tree admittance products.

2) The signal flowgraph method: The methods developed here are based on the idea proposed by Mason [8] in the 1950's. Formulation of the signal flowgraph and then the evaluation of the gain formula associated with it (Mason's formula) is the basis for symbolic analysis using this method. This method is used in the publicly available programs NASAP [9] and SNAP [10]. An improved signal flowgraph method which avoids term cancellations was described in [11].

3) The interpolation method: This method is best suited when the frequency variable s is the only symbolic variable in the network. It requires the finding of the coefficients of the network's determinant polynomial by evaluating it at different values of s [4]. However, using real values for s leads to ill-conditioned equations in addition to generating inaccurate solutions [12]. Therefore, it is best to use complex values for s. Some implementations of this method use Fast Fourier Transforms to find the coefficients of the determinant.

4) The parameter extraction method: This method was introduced in 1973 [13]. Other variations on the method were proposed later in [14] and [15]. The advantage of the method is that it is directly related to the basic determinant properties of widely used equation formulation methods like the modified nodal method and the tableau method.

The first generation of computer programs available for symbolic circuit simulation based on these methods include CORNAP [16], NASAP [9], and SNAP [10]. Research in the late 1980's and early 1990's has produced newer sym-

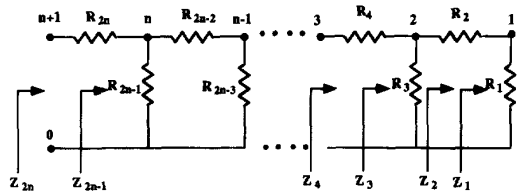


Fig. 1. Resistive ladder network.

bolic analysis programs. These programs include ISAAC [17], ASAP [18], EASY [19], BRAINS [20], and SSPICE [21]. These methods have network size limitations. The main problem is the exponential growth of the number of terms involved in the expression for the transfer function in (1) as the network gets larger. The solution to analyzing large-scale circuits lies in a total departure from the traditional procedure of trying to state the transfer function as a single expression and using a *sequence of expressions* procedure instead. The idea is to produce a succession of small expressions with a backward hierarchical dependency on each other. The growth of the number of expressions in this case will be shown to be linear for practical circuits (Section IV).

1.2. The Sequence of Expressions

The advantage of having the transfer function stated in a single expression lies in the ability to gain insight to the relationship between the transfer function and the network elements by inspection [4]. For large expressions though, this is not possible, and the single expression loses that advantage. ISAAC [22], ASAP [23], and SYNAP [24] attempt to handle larger circuits by maintaining the single expression method and using circuit dependent approximation techniques. The tradeoff is accuracy for insight. Therefore, the *sequence of expressions* approach is more suitable for accurately handling large-scale circuits. The following example illustrates the features of the *sequence of expressions*.

Example 1: Consider the resistance ladder network in Fig. 1. The goal is to obtain the input impedance function of the network, $Z_{\text{in}} = (V_{\text{in}}/I_{\text{in}})$. The single expression transfer function Z_4 is

$$Z_4 = \frac{R_4R_1 + R_4R_2 + R_4R_3 + R_3R_1 + R_3R_2}{R_1 + R_2 + R_3}. \quad (2)$$

The number of terms in the numerator and denominator are given by the Fibonacci numbers satisfying the following difference equation:

$$y(k+2) = y(k+1) + y(k) \qquad k = 0, 1, 2, \cdots$$
$$y(0) = 0,\ y(1) = 1. \quad (3)$$

An explicit solution to the above equation is [25]:

$$y(n) = \frac{1}{\sqrt{5}}\left[\left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{1-\sqrt{5}}{2}\right)^n\right], \qquad n \geq 0$$
$$\approx 0.447 \times 1.618^n \text{ for large } n \quad (4)$$

where n is the number of resistors in the ladder network.

The solution shows that the number of terms in Z_n increases exponentially with n. Any single expression transfer function has this inherent limitation.

Now using the *sequence of expressions* procedure, the input impedance can be obtained from the following expressions:

$$Z_1 = R_1 \quad (5)$$
$$Z_2 = R_2 + Z_1 \quad (6)$$
$$Z_3 = \frac{Z_2R_3}{Z_2 + R_3} \quad (7)$$
$$Z_4 = R_4 + Z_3. \quad (8)$$

For each additional resistance added the sequence of expressions will grow by one expression, either of the form $R_i + Z_{i-1}$ or $R_iZ_{i-1}/(R_i + Z_{i-1})$. The number of terms in the sequence of expressions can be given by

$$y(n) = \begin{cases} 2.5n - 2 & \text{for } n \text{ even} \\ 2.5n - 1.5 & \text{for } n \text{ odd} \end{cases} \quad (9)$$

which exhibits a linear growth with respect to n. So, to find the input impedance of a 100 resistor ladder network the single expression methods would produce 3.5×10^{20} terms which requires unrealistically huge computer storage capabilities. On the other hand, the *sequence of expressions* method would produce only 248 terms, which is even within the scope of some desk calculators.

Another advantage of the *sequence of expressions* is the number of mathematical operations needed to evaluate the transfer function. To evaluate Z_9, for example, the single expression method would require 302 multiplications and 87 additions. The *sequence of expressions* method would only require 8 multiplications and 8 additions, a large reduction in computer evaluation time. All this makes the concept of symbolic circuit simulation of large-scale networks very possible.

The *sequence of expressions* concept was first utilized in system analysis by Endy and Lin in 1981 [26]. The concept is also a basic part of compiler design [27]. However, its use is contingent upon the existence of the single expression which is then decomposed into a sequence of expressions. Its use in this paper is different, the sequence of expression is generated directly without needing or generating the final single expression. Furthermore, the single expression is not available nor is it possible to obtain for large-scale circuits because of the exponential nature of the size of that expression.

1.3. Hierarchical Methods

Two topological analysis methods for symbolic simulation of large-scale circuits have been proposed in [28] and in [29]. The first method utilizes the *sequence of expressions* idea to obtain the transfer functions. The method operates on the Coates flowgraph representing the circuit. A partitioning is proposed onto the flowgraph and not the physical network. The physical meaning of the partition is not inherently apparent. The process requires a flattening of the circuit, i.e., eliminating the hierarchy, in order to obtain the complete Coates graph and then a partitioning of the graph. This results in a loss of any predefined partitions (subcircuits). The second method

also utilizes the *sequence of expressions* and a Mason's signal flowgraph [8] representation of the circuit. The method makes use of partitioning on the physical level rather than on the graph level. Therefore, for a hierarchical circuit as is the case herein, the method can operate on the subcircuits in a hierarchical fashion in order to produce a final solution. Section V shows experimental comparisons with these two methods.

The network approach herein uses circuit partitioning which makes full use of predefined subcircuits. These are considered natural partitions and no flattening of the circuit is performed. This feature is most powerful for circuit design applications where a library of building blocks and their symbolic models can be established. This would conserve the computation time needed to re-analyze these subcircuits.

1.4. The Network Approach

The significance of the network approach described in this paper is the introduction of a noniterative hierarchical network approach to symbolic analysis. The algorithm consists of a combination of several well-known techniques for numerical circuit analysis and matrix manipulation. The method is aimed at analyzing large circuits in the size range of thousands of nodes. All previous symbolic methods proposed for large-scale circuits have been based on a flowgraph approach [28], [29]. The result is the production of a Symbolic Circuit Analysis Program with Partitioning (SCAPP). The algorithm proposes a noniterative solution to the partitioned subcircuits and consequently a recombination of these solutions hierarchically to produce the final solution to the complete network. One of the main issues that was kept in perspective is the utilization of multiprocessor computer systems [30]. This required the design of parallel algorithms that will allow the analysis to be performed concurrently on several parts of the circuit.

The analysis process is divided into the following parts: 1) Binary circuit partitioning; 2) subcircuit analysis, referred to as terminal block analysis; and 3) upward hierarchical analysis, referred to as middle block analysis.

First, a suitable network partitioning is performed (Fig. 2) during which a binary tree, modeling that process, is defined (Fig. 3). Each network subcircuit is represented by a leaf of the binary tree which also represents a terminal block analysis step. Each nonleaf vertex (parent) in the tree represents a binary partitioning operation in addition to a middle block analysis step. Terminal and middle block analysis steps are performed by upwardly traversing the binary tree until the root is reached.

The input to SCAPP is a netlist description of the circuit elements and their interconnections (the circuit topology). The elements allowed are resistors, capacitors, inductors, current and voltage controlled current and voltage sources, and ideal operational amplifiers (op amps). Semiconductor devices are handled as subcircuits using linear small signal models built around an initial dc solution to the circuit. SCAPP implements an automatic circuit partitioning algorithm in addition to allowing for user defined subcircuits [31].

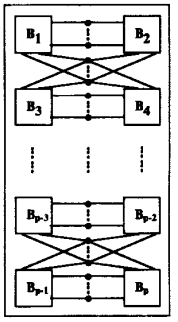


Fig. 2. General partitioning of a network.

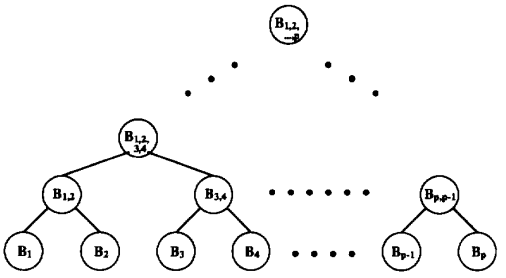


Fig. 3. Binary tree model of the partitioned network of Fig. 1.

Section II of this paper illustrates the terminal block analysis methodology and Section III details the hierarchical middle block analysis process. Section IV analyzes the resulting *sequence of expressions* and Section V shows the experimental results of SCAPP. Section VI lists the conclusions of this work.

II. Terminal Block Analysis

Terminal block analysis is the analysis of a circuit represented by an n-terminal block in order to characterize its electrical behavior. For this symbolic analysis methodology, the characterization is done in terms of the block's terminal node voltages and some extra branch currents, referred to simply as the subcircuit or network variables. A symbolic system of equations has to be extracted from the subcircuit in order to find the symbolic solution for the network variables. The Modified Nodal Analysis method (MNA) [32] is used to initially find the system of equations. The advantages of using MNA is that it allows branch currents as network variables in the analysis, which in turn leads to the ability to include voltage sources and all four types of controlled power sources. A Laplace Transform representation of the admittance value of the circuit elements is used in the formulation of its equations. The system of equations is therefore written as

$$\begin{bmatrix} \boldsymbol{Y_n} & \boldsymbol{B} \\ \boldsymbol{C} & \boldsymbol{D} \end{bmatrix} \begin{bmatrix} \boldsymbol{V} \\ \boldsymbol{I} \end{bmatrix} = \begin{bmatrix} \boldsymbol{J} \\ \boldsymbol{E} \end{bmatrix} \tag{10}$$

where $\boldsymbol{V}$ is the node-to-datum voltages, $\boldsymbol{I}$ is the branch current variables, $\boldsymbol{Y_n}$ is the modified nodal admittance matrix, $\boldsymbol{B}, \boldsymbol{C}$ and $\boldsymbol{D}$ are the contributions of the branch relationship equations, $\boldsymbol{J}$ represents the current sources in the network, and $\boldsymbol{E}$ represents the independent voltage sources.

The next step is suppressing all the internal variables to the subcircuit which mathematically means writing the system of (10) in terms of the external variables only (tearing node voltages and external branch current variables). This results in a huge reduction in the size of the matrices and the elimination of information not needed or requested by the analysis. The resulting equations are referred to as the Reduced Modified Nodal Analysis equations (RMNA). The procedure utilizes the concepts of Gaussian elimination [33] and Schur Complement [34] to reduce the matrices and generate the final transfer function. In order to illustrate how to formulate the RMNA system, (10) can be generically rewritten as:

$$\begin{bmatrix} \boldsymbol{M_I} & \boldsymbol{M_{IE}} \\ \boldsymbol{M_{EI}} & \boldsymbol{M_E} \end{bmatrix} \begin{bmatrix} \boldsymbol{X_I} \\ \boldsymbol{X_E} \end{bmatrix} = \begin{bmatrix} \boldsymbol{L_I} \\ \boldsymbol{L_E} \end{bmatrix}. \tag{11}$$

To suppress the internal variables $\boldsymbol{X_I}$ a new matrix $\boldsymbol{M_R}$ is given by

$$\boldsymbol{M_R} = \boldsymbol{M_E} - \boldsymbol{M_{EI}}(\boldsymbol{M_I})^{-1}\boldsymbol{M_{IE}}. \tag{12}$$

$\boldsymbol{L}$ is reduced similarly to $\boldsymbol{L_R}$. The new system of equations is given by

$$\boldsymbol{M_R X_E} = \boldsymbol{L_R}. \tag{13}$$

The process can be simplified a great deal if the internal variables are suppressed one at a time. Equation (12) reduces to

$$\boldsymbol{M_R} = \boldsymbol{M_E} - \frac{1}{a_{ii}}\boldsymbol{M_{Ei}}\boldsymbol{M_{iE}} \tag{14}$$

where $\boldsymbol{M_{Ei}}$ and $\boldsymbol{M_{iE}}$ are the column and the row for which a_{ii} (referred to as the pivot) is the intersection, respectively. Example 2 illustrates the terminal block analysis procedure.

Example 2: Consider the circuit labeled $\boldsymbol{B_1}$ in Fig. 4 with terminals 3 and 1 being the input and output terminals, respectively. Treating it as a 3-terminal block, nodes 1 and 3 become the external nodes. The assumption is that all current variables are to remain internal. Using element stamps [32], the MNA matrix for this circuit is expressed as

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ X_1 \end{array} \begin{array}{c} \begin{array}{cccc} v_1 & v_2 & v_3 & i_2 \end{array} \\ \begin{pmatrix} G_1 & -G_1 & 0 & 0 \\ -G_1 & G_1 + sC_3 & -sC_3 & 1 \\ 0 & -sC_3 & sC_3 + G_4 & 0 \\ 0 & 1 & -\mu_2 & 0 \end{pmatrix} \end{array}. \tag{15}$$

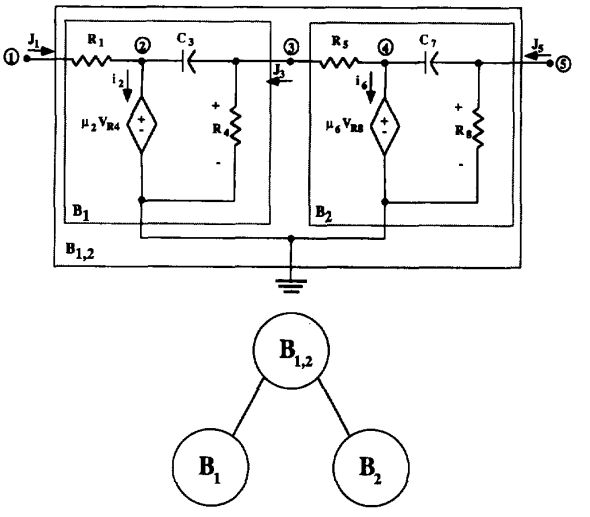


Fig. 4. A simple circuit and its hierarchical model.

To produce the RMNA matrix in terms of v_1 and v_3 only, all the other variables must be suppressed. The anticipated result is a 2 × 2 RMNA matrix. Notice that an attempt to reduce the current variable i_2 first would cause a problem because the pivot is zero. Therefore, its suppression is deferred. The reasoning behind the order of variable reduction is fully explained in Section 3.3. So, the first step here is to suppress v_2. The *pivot* is $G_1 + sC_3$. Applying (14) to the MNA matrix of (15) yields (16) (shown at the bottom of the page). Notice that the suppression of node 2, which is one of the nodes of the branch whose current is a variable, produced a fill in *pivot* position for row X_1. This now allows for the suppression of i_2. Equation (14) is applied again and results in

$$\begin{bmatrix} \dfrac{G_1 sC_3}{G_1 + sC_3} & -\dfrac{G_1 sC_3}{G_1 + sC_3} \\ -\dfrac{G_1 sC_3}{G_1 + sC_3} & \dfrac{G_1 sC_3}{G_1 + sC_3} + G_4 \end{bmatrix} + (G_1 + sC_3) \cdot \begin{bmatrix} \dfrac{G_1}{G_1 + sC_3} \\ \dfrac{sC_3}{G_1 + sC_3} \end{bmatrix} \begin{bmatrix} \dfrac{G_1}{G_1 + sC_3} & -\mu_2 + \dfrac{sC_3}{G_1 + sC_3} \end{bmatrix}. \tag{17}$$

$$\begin{bmatrix} G_1 & 0 & 0 \\ 0 & sC_3 + G_4 & 0 \\ 0 & -\mu_2 & 0 \end{bmatrix} - \frac{1}{G_1 + sC_3} \begin{bmatrix} -G_1 \\ -sC_3 \\ 1 \end{bmatrix} \begin{bmatrix} -G_1 & -sC_3 & 1 \end{bmatrix}$$

$$= \begin{array}{c} \\ 1 \\ 3 \\ X_1 \end{array} \begin{array}{c} \begin{array}{ccc} v_1 & v_3 & i_2 \end{array} \\ \begin{pmatrix} \dfrac{G_1 sC_3}{G_1 + sC_3} & -\dfrac{G_1 sC_3}{G_1 + sC_3} & \dfrac{G_1}{G_1 + sC_3} \\ -\dfrac{G_1 sC_3}{G_1 + sC_3} & \dfrac{G_1 sC_3}{G_1 + sC_3} + G_4 & \dfrac{sC_3}{G_1 + sC_3} \\ \dfrac{G_1}{G_1 + sC_3} & -\mu_2 + \dfrac{sC_3}{G_1 + sC_3} & -\dfrac{1}{G_1 + sC_3} \end{pmatrix} \end{array}. \tag{16}$$

The final RMNA matrix characterizing the 3-terminal block becomes

$$\begin{matrix}1\\3\end{matrix}\begin{bmatrix}G_1 & -G_1\mu_2\\ 0 & sC_3+G_4-sC_3\mu_2\end{bmatrix}\begin{bmatrix}v_1\\v_3\end{bmatrix}=\begin{bmatrix}J_1\\J_3\end{bmatrix} \quad (18)$$

where J_1 and J_3 are the currents entering terminals 1 and 3 from the external world.

For the case where ideal opamps exist in the circuit, a slightly different set of rules are used. A nullor model is used to describe the opamp [35]. The existence of the opamp simply affects the symmetry of the reduction process for opamp output nodes. The row in the MNA or RMNA matrix (KCL) corresponding to the output node is never built. The extra constraint equation replacing the output node KCL is the input constraint which states that the voltage across the input terminals is zero. When reduction of an opamp output node is to be processed, the row corresponding to the negative terminal input KCL and the column corresponding to the output node are suppressed. References [35] and [36] provide detailed descriptions of the process.

One of the major features of the matrices in this network approach is their extreme sparsity. For a circuit with approximately 1000 nodes its MNA matrix will be only about 0.8% full [35]. Partitioning and sparse matrix manipulation techniques are implemented in SCAPP to take advantage of this feature.

III. The Hierarchical Middle Block Analysis

Middle block analysis is the process of combining the electrical characterization of two n-terminal blocks to produce a characterization for a new n-terminal block which physically is the interconnection of the initial two blocks. Fig. 4 illustrates the idea of middle block analysis. Blocks $\boldsymbol{B_1}$ and $\boldsymbol{B_2}$ are both n-terminal blocks that will be characterized by two separate RMNA matrices, $\boldsymbol{M_R^1}$ and $\boldsymbol{M_R^2}$, respectively. The new n-terminal block $\boldsymbol{B_{1,2}}$ that results from the interconnection of blocks $\boldsymbol{B_1}$ and $\boldsymbol{B_2}$ will be characterized by a new RMNA matrix $\boldsymbol{M_R^{1,2}}$. The process of producing $\boldsymbol{M_R^{1,2}}$ is the middle block analysis.

3.1. Middle Block Analysis Model

Consider the general interconnection of a group of n-terminal blocks shown in Fig. 2. The n-terminal blocks represent the final subcircuits which are the result of the partitioning process, manual or automatic. Each one of these blocks is referred to as a *terminal* block. Fig. 3 shows the binary tree that models the binary partitioning process. The tree has p leaves, each corresponding to one of the terminal blocks. The partitioning uses a node tearing technique [31]; therefore some terminal blocks share common nodes. These tearing nodes are referred to as *external* nodes. Appropriately, the nodes local to a block are referred to as *internal* nodes. Combining two of the terminal blocks, as illustrated in Fig. 4, produces a new n-terminal block. This block is referred to as a *middle* block. After the characterization of the middle block has been completed, it is treated as a terminal block. A better understanding of the concept can be achieved by studying the middle block analysis of the two terminal blocks in Fig. 4.

In order to establish the terminology used in describing a block, *terminal* or *middle*, a typical n-terminal block is considered: block $\boldsymbol{B_i}$. $\boldsymbol{N^i}$ is the set of all the nodes of block i. It is defined as follows:

$$\boldsymbol{N^i}=\boldsymbol{N_I^i}\bigcup\boldsymbol{N_E^i} \quad (19)$$

where $\boldsymbol{N_I^i}$ is the set of *internal* nodes to the block, and $\boldsymbol{N_E^i}$ is the set of *external* nodes. There are n^i total nodes in $\boldsymbol{N^i}$. n^i is defined as

$$n^i=n_I^i+n_E^i \quad (20)$$

where n_I^i is the number of elements in $\boldsymbol{N_I^i}$, and n_E^i is the number of elements in $\boldsymbol{N_E^i}$. The complete set of tearing nodes for the entire circuit is $\boldsymbol{TN}$. The tearing nodes between two or more blocks are always a subset of $\boldsymbol{TN}$. This subset, $\boldsymbol{TN^{i,j,k\cdots}}\subset\boldsymbol{TN}$ is defined as follows:

$$\boldsymbol{TN^{i,j,k\cdots}}=\boldsymbol{N_E^i}\bigcap\boldsymbol{N_E^j}\bigcap\boldsymbol{N_E^k}\bigcap\cdots \qquad i\neq j\neq k. \quad (21)$$

For the simple example of Fig. 4,

$$\begin{aligned}\boldsymbol{N^1}&=\{0,1,2,3\} & n^1&=4 & n_E^1&=3 & n_I^1&=1\\ \boldsymbol{N_E^1}&=\{0,1,3\} & \boldsymbol{N_I^1}&=\{2\}\\ \boldsymbol{N^2}&=\{0,3,4,5\} & n^2&=4 & n_E^2&=3 & n_I^2&=1\\ \boldsymbol{N_E^2}&=\{0,3,5\} & \boldsymbol{N_I^2}&=\{4\}\\ \boldsymbol{TN^{1,2}}&=\{0,3\} & \boldsymbol{TN}&=\{0,1,3,5\}.\end{aligned}$$

The process of performing a symbolic analysis on a network follows the following steps.

1) Partitioning the circuit (or using user-defined subcircuits, which is the case for large-circuits) into the terminal blocks $\boldsymbol{B_i}(1<i<p)$ and obtaining the binary tree model shown in Fig. 3.
2) Performing the terminal block analysis on each block $\boldsymbol{B_i}$ (on each leaf of the binary tree) and obtaining a RMNA matrix $\boldsymbol{M_R^i}$ characterizing each one.
3) Successively performing the middle block analysis in a hierarchical binary fashion by traversing the partitioning binary tree upwards starting with the leaves until the root is reached. The final result is one RMNA matrix characterizing the entire network in terms of the defined input and output variables specified by the user. The algorithm is not limited to a hierarchical binary process only. The steps illustrated in Section 3.2 are intuitively expandable to any number of partitions on a given hierarchical level.

3.2. Middle Block Analysis

After steps 1 and 2 are performed, the network in Fig. 2 is characterized by p RMNA matrices in terms of the variables that are the members of the set $\boldsymbol{TN}$ in addition to the extra current variables requested from the analysis. Consider the

connection of any two terminal blocks, namely blocks B_i and B_j. The RMNA equations describing each are

$$[M_R^i] \cdot \begin{bmatrix} V_e^i \\ I_e^i \end{bmatrix} = \begin{bmatrix} J_e^i \\ 0 \end{bmatrix} \tag{22}$$

$$[M_R^j] \cdot \begin{bmatrix} V_e^j \\ I_e^j \end{bmatrix} = \begin{bmatrix} J_e^j \\ 0 \end{bmatrix}. \tag{23}$$

The two blocks share the tearing nodes in the set $TN^{i,j} = TN^i \bigcap TN^j$. Also $N_E^{i,j} = N_E^i \bigcup N_E^j$.

The following middle block analysis steps are followed in order to produce the RMNA matrix corresponding to a middle block.

1) Combine the vector of external variables for the two blocks

$$V^{i,j} = V_e^i \bigcup V_e^j, \tag{24}$$

$$I^{i,j} = I_e^i \bigcup I_e^j, \tag{25}$$

where the vectors V and I are the node voltages and current variables of the middle block $B_{i,j}$.

2) Combine the RMNA matrices to produce a temporary intermediate matrix M_{R_t} such that

$$M_{R_t} = \begin{bmatrix} M_R^i & 0 \\ 0 & M_R^j \end{bmatrix} \tag{26}$$

and also to produce the following temporary right hand side vector RHS_t

$$RHS_t = \begin{bmatrix} J_e^i \\ 0 \\ J_e^j \\ 0 \end{bmatrix}. \tag{27}$$

3) Add the columns and rows of M_{R_t} that correspond to the tearing node voltage variables shared between the two blocks, that is, the nodes that are members of the set $TN^{i,j}$. Mathematically, adding the columns in this step reflects the fact that the voltage at a tearing node is equal in all blocks. The concept of adding the rows is not as simple. Each row corresponding to a tearing node represents the sum of the currents entering the block at that node. The corresponding entry in the RHS vector symbolizes the current entering that node from the rest of the network. A tearing node shared between the two blocks falls under one of three categories.

   3.1) The tearing node is only an element of N_E^i and N_E^j. So it is a local tearing node for block $B_{i,j}$ and therefore is an internal node to $B_{i,j}$ and a member of the set N_I^i.

   3.2) The tearing node is also an element of N_E^k $k \neq i$ and $k \neq j$. That is, it is shared by more than the blocks B_i and B_j. This dictates that it must remain as a tearing node, ie. its row and column not be suppressed. It is therefore an external node to $B_{i,j}$ and a member of the set $N_E^{i,j}$.

   3.3) The tearing node voltage variable is requested by the analysis in which case it becomes a member N_E^i.

   So, when adding the rows of a node corresponding to 3.1 above, the entry in RHS_t will become zero. This is because the current entering block B_i from the rest of the network through the tearing node (only B_j in this case) is equal to the negative of the current entering block B_j from the rest of the network (only B_i in this case). However, when adding the rows of a node corresponding to steps 3.2 and 3.3 above, the entry in $J_e^{1,2}$ becomes simply a symbol indicating the current entering that node from the rest of the network (outside $B_{i,j}$.

4) Suppress all the internal variables to block $B_{i,j}$ in order to obtain the RMNA matrix characterizing the block in terms of its external variables. After step 3 is completed the sets $N_I^{i,j}$ and $N_E^{i,j}$ become completely defined and the intermediate RMNA system of equations can be written as

$$[M_{R_t}^{i,j}] \cdot \begin{bmatrix} V^{i,j} \\ I^{i,j} \end{bmatrix} = RHS_t. \tag{28}$$

The suppression of the tearing nodes shared between the two blocks and corresponding to case 3.1 above will reduce the $V^{i,j}$ vector to the external voltage variable vector $V_e^{i,j}$. To understand the entries of $I^{i,j}$ a look at how a current variable reaches a middle block analysis stage is considered. A branch current variable is always an internal variable unless is has been requested by the analysis or has had a pivoting problem and was not reduced earlier, in which case it has been labeled as an *external* variable. Conceptually though it can never be an *external* variable. Blocks do not share branch currents, they share node voltages; because the partitioning is a node tearing technique. Therefore, an attempt to reduce a current variable that has not been requested by the analysis must be made. The current variables considered here all have a pivot problem. The pivot problem might have been resolved by the creation of a fill at that position because of the suppression of the now internal nodes that control the pivot of the current variable. If not, the variable is not suppressed and its row and column are carried along further up the binary tree. So, $I_e^{i,j} = I^{i,j}$ unless the suppression of one or more "healed" variables was successful, in which case $I_e^{i,j} \subset I^{i,j}$.

The final result of the middle block analysis of blocks B_i and B_j is the RMNA matrix characterizing middle block $B_{i,j}$ in terms of its external variables. The RMNA system

of equations is written as

$$[\boldsymbol{M}_{\boldsymbol{R}}^{\boldsymbol{i,j}}] \cdot \begin{bmatrix} \boldsymbol{V}_{\boldsymbol{e}}^{\boldsymbol{i,j}} \\ \boldsymbol{I}_{\boldsymbol{e}}^{\boldsymbol{i,j}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{J}_{\boldsymbol{e}}^{\boldsymbol{i,j}} \\ \boldsymbol{0} \end{bmatrix}. \tag{29}$$

The current variables are grouped at the bottom of the variable vector strictly for cosmetic reasons and has no effect on the analysis. An example will serve to illustrate the above middle block analysis steps.

Example 3: Consider the circuit in Fig. 4 with the input terminal 5 and the output terminal 1. The circuit is partitioned into two terminal blocks: $\boldsymbol{B_1}$ and $\boldsymbol{B_2}$ and the binary tree in Fig. 4 models the analysis. The terminal block analysis of each block was performed in example 2. Notice that $\boldsymbol{TN}^{i,j} = \{3\}$ and node 3 is local to the middle block $\boldsymbol{B_{1,2}}$. Therefore $\boldsymbol{N_I^{1,2}} = \{3\}$ and $\boldsymbol{N_E^{1,2}} = \{1,5\}$. Concatenating $\boldsymbol{M_R^1}$ and $\boldsymbol{M_R^2}$ and adding the rows and columns corresponding to the tearing node results in (30) (shown at the bottom of the page).

The final step of the middle block analysis is to delete the row and column corresponding to the internal variable v_3. The result is the following RMNA matrix characterizing the middle block $\boldsymbol{B_{1,2}}$:

$$\boldsymbol{M_R^{1,2}} = \begin{array}{c} \\ 1 \\ 5 \end{array} \begin{array}{c} \begin{array}{cc} v_1 & v_5 \end{array} \\ \begin{bmatrix} G_1 & \dfrac{-G_1\mu_2 G_5\mu_6}{sC_3 + G_4 - sC_3\mu_2 + G_5} \\ 0 & sC_7 + G_8 - sC_7\mu_6 \end{bmatrix} \end{array}. \tag{31}$$

3.3. The Hierarchical Analysis

The process of recursively applying the above middle block analysis up the binary tree of Fig. 3 at each nonleaf vertex (middle block) will produce one final RMNA matrix characterizing the network in terms of the variables requested by the analysis. The terminal block analysis is performed on the leaves of the tree and p RMNA matrices are produced. At the second level of the binary tree, the middle block analysis is performed on pairs of terminal blocks. The process will produce $p/2$ RMNA matrices representing the $p/2$ middle blocks. As illustrated earlier, physically a middle block is a subcircuit. It is the result of interconnecting two closely coupled smaller subcircuits. So terminating the hierarchical analysis at this point will yield a characterization of the $p/2$ subcircuits represented by the middle blocks. At this point the leaves are no longer of any use to the process. They may be deleted after the middle block analysis at their parent vertex has been performed. The $p/2$ middle blocks and their RMNA characterizations become the leaves of the new reduced binary tree. They can be treated as terminal blocks and the middle block analysis repeated at the next level. The process is a recursive one and can be illustrated by the following function:

```
analyze(parent) /* Recursively Manage the terminal and
        Middle block analyzes */
{
 if (no left_child) { /* If there is a left child then a
                        right one must exist */
 analyze (left_child);
 analyze (right_child);
 middle (parent); /* Perform middle block analysis */
 }
 else {      /* A leaf has been reached */
 terminal(parent); /* Perform terminal block analysis */
 }
}
```

The analysis process is started by a call to the recursive function *analyze(root)* at the top of the binary tree. The function will traverse the binary tree in an *postorder* fashion. The functions *terminal()* and *middle()* return RMNA matrices to *analyze()*.

The structure of the binary tree is dictated by the partitioning (Fig. 3). So depending on the partitioning, the structure of the binary tree will range between a balanced binary tree and a totally unbalanced binary tree. A balanced binary tree is a highly desirable structure especially for the case where the hierarchical analysis is performed on a multiprocessor computer. The algorithm is very parallelizable because each terminal block analysis is a totally independent process and each middle block analysis is only dependent on its children. Examining Fig. 3 will show that block $\boldsymbol{B_{1,2,3,4}}$ has to wait for blocks $\boldsymbol{B_{1,2}}$ and $\boldsymbol{B_{3,4}}$ to finish before it can be processed, which in turn requires all four blocks $\boldsymbol{B_1}, \boldsymbol{B_2}, \boldsymbol{B_3}$, and $\boldsymbol{B_4}$ to finish. This is the hierarchical dependency of the process. The vertices on the same level are always totally independent processes.

IV. Analysis of the Sequence of Expressions

The quality of the results of the analysis are measured by statistics based on the resultant *sequence of expressions* from SCAPP. A good symbolic result for a large circuit is one that has a minimal number of terms and produces a minimal number of mathematical operations. The latter is very important, since the main usage of this type of symbolic analysis is the repetitive evaluations of the results in order to characterize a circuit and optimize circuit element values. This section will show that the number of symbolic terms produced by SCAPP grows linearly as a function of the size of the circuit (for practical circuits) and finds an estimate for the number of operations (multiplications, divisions, additions and subtractions) generated in the *sequence of expressions*.

$$\boldsymbol{M}_{\boldsymbol{R_t}} = \begin{array}{c} \\ 1 \\ 3 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccc} v_1 & v_3 & v_5 \end{array} \\ \begin{pmatrix} G_1 & -G_1\mu_2 & 0 \\ 0 & sC_3 + G_4 - sC_3\mu_2 + G_5 & -G_5\mu_6 \\ 0 & 0 & sC_7 + G_8 - sC_7\mu_6 \end{pmatrix} \end{array} \tag{30}$$

The basic terminal analysis process is given by (14). The actual implementation is performed on an element by element basis of the MNA and RMNA matrices. The equation for updating the RMNA matrix element a_{jk} as a result of reducing the ith variable (ith row and column), is given by

$$\text{new } a_{jk} = a_{jk} - \frac{1}{a_{ii}} * a_{ji} * a_{jk} \qquad \text{for } j \neq i \text{ and } k \neq i. \quad (32)$$

The process is performed on the matrix, row by row, only for rows where element $a_{ji} \neq 0$. Therefore, the term

$$p_i = \frac{1}{a_{ii}} * a_{ji} \quad (33)$$

need only be generated once for each row. Equation (32) can then be written as

$$\text{new } a_{jk} = a_{jk} - p_i * a_{ik}. \quad (34)$$

Equation (34) is generated for each element in row j for which $a_{ik} \neq 0$. The goal here is to deal with "real" circuits. What this means is that the number of branches incident on a node are bound by a constant, K, which is on the order of 6 in the worse case (excluding ground and power supply nodes). This results in the fact that the number entries in a row of an MNA matrix is bound by the constant $K_1 = 2K + L$, where L is a constant on the order of 2 in the worse case to account for the matrix skew due to dependent sources in the circuit [32]. By the same analysis, the number of entries in a column is given by a constant K_2. So for a worse case analysis, assuming no partitioning, the process of reducing an MNA matrix with n variables ($n \times n$ matrix) to an RMNA matrix with 2 variables (2×2) matrix, will generate a total number of symbols

$$N_{\text{sym}} \approx nK_0 + \sum_{i=1}^{n-2} k_2(3 + 4K_1) \quad (35)$$

where the $(3+4K_1)$ is the total number of symbols generated from the processing of one row, (33) and (34), and (nK_0) is the number of symbols needed to represent the initial entries of the MNA matrix. Equation (35) can be reduced to $(nK_0) + (n-2)K_2(3+4K_1)$ which is $\boldsymbol{O(n)}$. The K's in the above equation are kept constant because of the fact that some fills are created in the matrix as the size of the matrix decreases. A good partitioning algorithm (or a reordering algorithm) can guarantee a nonincreasing K_1 and K_2 [35].

The number of operations generated in the sequence of expressions is given by

$$N_{\text{mults}} \approx K_3 + \sum_{i=1}^{n-2} K_2(1 + K_1)$$
$$= K_3 + (n-2)(K_2 + K_1K_2) \quad (36)$$
$$N_{\text{adds}} \approx K_4 + \sum_{i=1}^{n-2} K_2 = K_4 + (n-2)K_2 \quad (37)$$

where N_{mults} represents the number of multiplications and divisions, N_{adds} represents the number of additions and subtractions, and K_3 and K_4 represent the operations needed to generate the initial entries of the MNA matrix (usually only the diagonal will need some addition operations and K_3 represents the frequency variable multiplications).

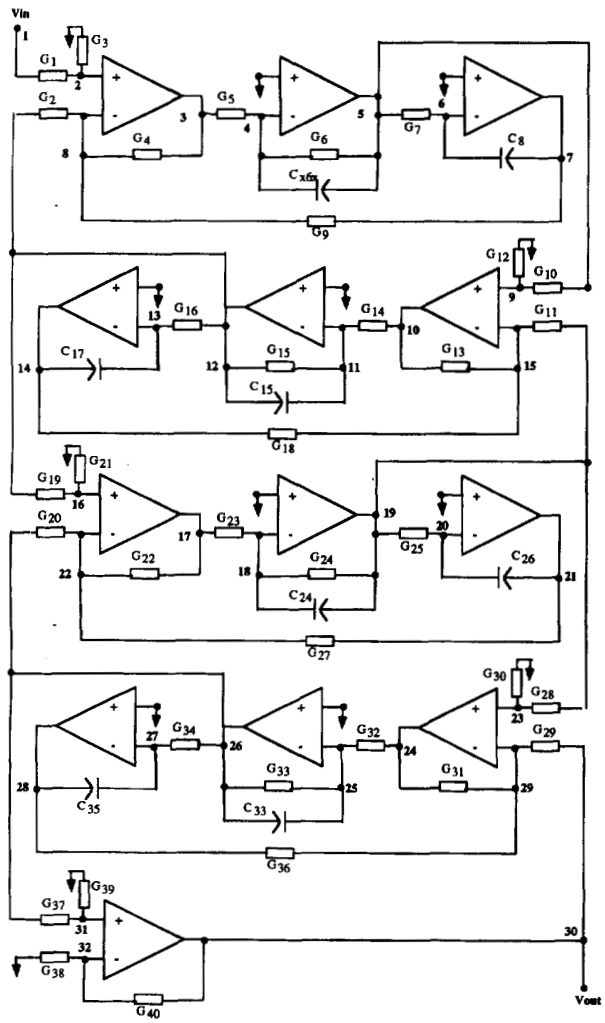


Fig. 5. Band-pass filter [28].

TABLE I
RESULTS FROM BAND-PASS FILTER ANALYSIS

Partitioning Status	Analysis Method	Band-Pass Filter		
		Muls	Adds	Eqs
With Partitioning	Coates (flowgraph)	63	30	25
	SCAPP (network)	48	27	56
	MASSAP (flowgraph)	70	29	60
NO Partitioning	Coates (flowgraph)	**	**	**
	SCAPP (network)	79	35	88
	MASSAP (flowgraph)	89	38	85

* Number of multiplications include divisions and number of additions include subtractions.
** Data is not available for the unpartitioned case.

V. EXPERIMENTAL RESULTS

Three sets of experimental results are presented. The first is the analysis of a band-pass filter circuit (Fig. 5) that was simulated using the other two hierarchical symbolic approaches that use *sequence of expressions:* the Coates flowgraph approach [28] and the Mason's signal flowgraph approach (MASSAP) [29]. The results are presented in Table I in terms of the efficiency of the output expressions. For the partitioned results, the Coates flowgraph approach performs partitioning on the

TABLE II
SCAPP RESULTS FOR BAND-PASS FILTER ($V_{out}/V_{in} = -T_0(32,1)/T_0(32,30)$)

TERMINAL BLOCKS		
Block Number	**Sym Function**	**Symbolic Expression**
5 TN= {26,30}	P(1)	$G_{37}+G_{39}$
	P1(32)	$(G_{38}+G_{40})/P(1)$
	T1(32,26)	$-P1(32)*(-G_{37})$
4 TN = {19,26,30}	P(3)	$G_{28}+G_{30}$
	P3(29)	$(G_{29}+G_{31}+G_{36})/P(3)$
	T3(29,19)	$-P3(29)*(-G_{28})$
	P3(25)	$(-G_{32})/(-G_{31})$
	T3(25,30)	$-P3(25)*(-G_{29})$
	T3(25,19)	$-P3(25)*T3(29,19)$
	T3(25,28)	$-P3(25)*(-G_{36})$
	P3(25)	$T3(25,28)/(-sC_{35})$
	T3(25,26)	$(-G_{33}-sC_{33})$ $-P3(25)*(-G_{34})$
3 TN = {12,19,26}	P(5)	$G_{19}+G_{21}$
	P5(22)	$(G_{20}+G_{22}+G_{27})/P(5)$
	T5(22,12)	$-P5(22)*(-G_{19})$
	P5(18)	$(-G_{23})/(-G_{22})$
	T5(18,12)	$-P5(18)*T5(22,12)$
	T5(18,26)	$-P5(18)*(-G_{20})$
	T5(18,21)	$-P5(18)*(-G_{27})$
	P5(18)	$T5(18,21)/(-sC_{26})$
	T5(18,19)	$(-G_{24}-sC_{24})$ $-P5(18)*(-G_{25})$
2 TN = {5,12,19}	P(7)	$G_{10}+G_{12}$
	P7(15)	$(G_{11}+G_{13}+G_{18})/P(7)$
	T7(15,5)	$-P7(15)*(-G_{10})$
	P7(11)	$(-G_{14})/(-G_{13})$
	T7(11,5)	$-P7(11)*T7(15,5)$
	T7(11,19)	$-P7(11)*(-G_{11})$
	T7(11,14)	$-P7(11)*(-G_{18})$
	P7(11)	$T7(11,14)/(-sC_{17})$
	T7(11,12)	$-G_{15}-sC_{15}$ $-P7(11)*(-G_{16})$
1 TN = {1,5,12}	P(8)	G_1+G_3
	P8(1)	$(-G_1)/P(8)$
	T8(1,1)	$(G_1)-P8(1)*(-G_1)$
	P8(8)	$(G_2+G_4+G_9)/P(8)$
	T8(8,1)	$-P8(8)*(-G_1)$
	P8(4)	$(-G_5)/(-G_4)$
	T8(4,1)	$-P8(4)*T8(8,1)$
	T8(4,12)	$-P8(4)*(-G_2)$
	T8(4,7)	$-P8(4)*(-G_9)$
	P8(4)	$T8(4,7)/(-sC_8)$
	T8(4,5)	$-G_6-sC_6-P8(4)*(-G_7)$

MIDDLE BLOCKS		
Block Number	**Sym Function**	**Symbolic Expression**
6 TN = {1,12,19}	P6(11)	$T7(11,5)/T8(4,5)$
	T6(11,1)	$-P6(11)*T8(4,1)$
	T6(11,12)	$T7(11,12)$ $-P6(11)*T8(4,12)$
7 TN = {1,19,26}	P4(18)	$T5(18,12)/(T6(11,12)$
	T4(18,1)	$-P4(18)*T6(11,1)$
	T4(18,19)	$T5(18,19)$ $-P4(18)*T7(11,19)$
8 TN = {1,26,30}	P2(25)	$T3(25,19)/(T4(18,19)$
	T2(25,1)	$-P2(25)*T4(18,1)$
	T2(25,26)	$T3(25,26)$ $-P2(25)*T5(18,26)$
9 TN = {1,30}	P0(32)	$T1(32,26)/T2(25,26)$
	T0(32,1)	$-P0(32)*T2(25,1)$
	T0(32,30)	$-G_{40}$ $-P0(32)*T3(25,30)$

signal flowgraph rather than on the circuit. SCAPP and MASSAP make use of circuit partitioning thus exploiting naturally hierarchical circuits. Fig. 6 shows the basic building blocks of the circuit, Fig. 7 shows the hierarchical interconnection of these blocks, and Fig. 8 shows the binary tree model. Table II shows the resulting *sequence of expressions* output from SCAPP for the partitioned circuit.

The second set of results represent some comparisons with MASSAP that illustrate the difference in performance for two classes of circuit configurations. Both methods can exploit circuit partitioning; however, the results show the methods to be comparable for circuits where the Mason's signal flowgraph of a circuit has a one-to one branch correspondence with circuit elements, namely ideal opamp circuits [29]. Table III shows that SCAPP results are more efficient for cases where the branches in MASSAP do not exhibit a one-to-one correspondence between circuit and flowgraph, which is usually the case.

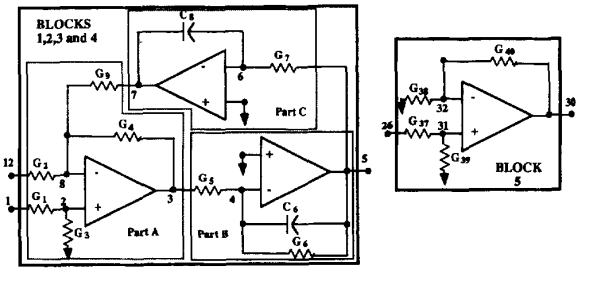


Fig. 6. Building blocks of the band-pass filter.

The third set of data illustrate an interesting effect partitioning has on the network approach in SCAPP. Table IV

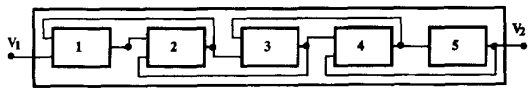


Fig. 7. Partitioned band-pass filter.

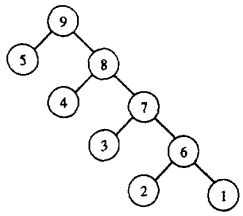


Fig. 8. Analysis model for the band-pass filter.

TABLE III
COMPARISON OF MASSAP WITH SCAPP*

Analysis Method	Figure 6 Basic Cell			Figure 9 Biquad [37]			Figure 1 Ladder Network (10 Nodes)		
	Mults	Adds	Eqs	Mults	Adds	Eqs	Mults	Adds	Eqs
SCAPP	16	7	23	16	6	22	48	34	54
MASSAP	13	5	11	16	5	18	67	33	67

* Number of multiplications include divisions and number of additions include subtractions

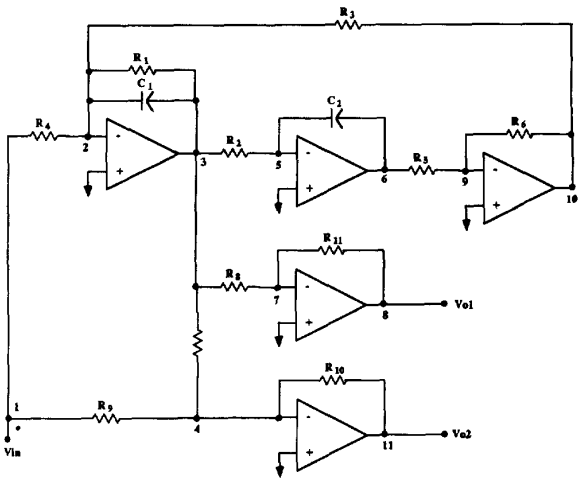


Fig. 9. Biquad circuit [37].

shows the results of SCAPP runs for the circuit in Fig. 10. The second column indicates the order in which internal variables were suppressed. The first row shows the results with the automatic partitioner invoked on the circuit. This produces a near-optimal scheduling of the internal variables to be suppressed by SCAPP. The result is a minimal number of fills in the RMNA matrices of the partitioned and recombined subcircuits. Rows 2 and 3 of Table IV show the results with no partitioning performed and a random scheduling of variables to be suppressed. This scheduling has a great effect on the number of fills in the RMNA matrix for the circuit, as can be seen from the last column in Table IV. An optimal scheduling can be obtained by a permutation of the rows and columns of the MNA or RMNA matrices, which can achieve the results obtained by using partitioning. This process, however,

TABLE IV
RESULTS FROM SCAPP ANALYSIS OF FIG. 10 CIRCUIT

Type of Partitioning	Reduction Order	Number of multiplications	Number of additions	Number of equations	Diff* %
auto	auto	66	70	75	-
none	random	287	216	292	76
	random	102	89	108	34

* A multiplication is considered equivalent to 6 additions for a typical computer.

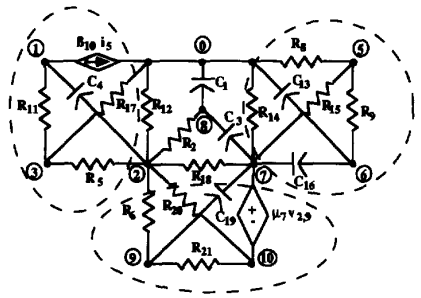


Fig. 10. Automatically partitioned circuit.

would require the flattening of hierarchical circuits. Therefore, partitioning rather than matrix reordering makes use of the natural hierarchy of large circuits and eliminates the need to flatten naturally hierarchical circuit.

VI. CONCLUSION

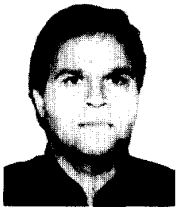
A new method for the symbolic analysis of large-scale circuits has been presented. The method features a hierarchical network approach to the problem that consists of two parts: the terminal block analysis and the middle block analysis. Although a hierarchical method, the algorithm is capable of performing the analysis on an entire circuit without the need for partitioning, which allows for the flexibility in using partitioning. The method facilitates the characterization and building of standard symbolic libraries for standard circuit building blocks. The partitioning and analysis procedures are capable of handling these blocks as user-defined subcircuits which results in a more efficient analysis process. Each terminal block analysis result is a complete solution of the subcircuit in terms of its external variables. Another powerful feature of the analysis is its parallel processing properties. It is very suitable for implementation on multiprocessor machines [30]. Immediate future work includes the development of hierarchical symbolic sensitivity analysis techniques and the extension of the algorithm to time domain analysis.

The source code for SCAPP and its users manual are available for release through requests to the first author (e-mail: <marwan@iastate.edu>). The program is written in *C* and is supported on a UNIX™ workstation platform.

REFERENCES

[1] G. Gielen and W. Sansen, *Symbolic Analysis for Automated Design of Analog Integrated Circuits*. Norwell, MA: Kluwer, 1991.
[2] P. M. Lin, "Sensitivity analysis of large linear networks using symbolic programs," in *IEEE Int. Symp. Circuits Syst.*, San Diego, CA, May 1992, pp. 1145–1148.
[3] A. Konczykowska, M. Bon, H. Wang, and R. Mezui-Mintsa, "Symbolic analysis as a tool for circuit optimization," in *IEEE Int. Symp. Circuits Syst.*, San Diego, CA, May 1992, pp. 1161–1164.

[4] P. M. Lin, *Symbolic Network Analysis*. Ansterdam: Elsevier Science, 1991.
[5] D. A. Calahan, "Linear network analysis and realization digital computer programs, and instruction manual," *University of Ill. Bull.*, vol. 62, Feb. 1965.
[6] J. O. McClanahan and S. P. Chan, "Computer analysis of general linear networks using digraphs," *Int. J. Electron.*, no. 22, pp. 153–191, 1972.
[7] L. O. Chua and P. M. Lin, *Computer Aided Analysis of Electronic Circuits—Algorithms and Computational Techniques*. Englewood Cliffs, NJ: Prentice–Hall, 1975.
[8] S. J. Mason, "Feedback theory—Further properties of signal flow graphs," *IRE*, vol. 44, pp. 920–926, July 1956.
[9] L. P. McNamee and H. Potash, "A user's and programmer's manual for NASAP," University of California at Los Angeles, Rep. 63-38, Aug. 1968.
[10] P. M. Lin and G. E. Alderson, "SNAP—A computer program for generating symbolic network functions," School of EE, Purdue Univ., West Lafayette, IN, Rep. TR-EE 70-16, Aug. 1970.
[11] R. R. Mielke, "A new signal flowgraph formulation of symbolic network functions," *IEEE Trans. Circuits Syst.*, vol. CAS-25, pp. 334–340, June 1978.
[12] K. Singhal and J. Vlach, "Generation of immittance functions in symbolic form for lumped distributed active networks," *IEEE Trans. Circuits Syst.*, vol. CAS-21, pp. 57–67, Jan. 1974.
[13] G. E. Alderson and P. M. Lin, "Computer generation of symbolic network functions-A new theory and implementation," *IEEE Trans. Circuit Theory*, vol. CT-20, pp. 48–56, Jan 1973.
[14] K. Singhal and J. Vlach, "Symbolic analysis of analog and digital circuits," *IEEE Trans. Circuits Syst.*, vol. CAS-24, pp. 598–609, Nov. 1977.
[15] P. Sannuti and N. N. Puri, "Symbolic network analysis—An algebraic formulation," *IEEE Trans. Circuits Syst.*, vol. CAS-27, pp. 679–687, Aug. 1980.
[16] C. Pottle, CORNAP User Manual, School of Elec. Eng., Cornell Univ., Ithaca, NY, 1968.
[17] G. Gielen, H. Walscharts, and W. Sansen, "ISAAC: A symbolic simulator for analog integrated circuits," *IEEE J. Solid-State Circuits*, vol. SC-24, pp. 1587–1597, Dec. 1989.
[18] F. V. Fernández, A. Rodríguez-Vázquez, and J. L. Huertas, "An advanced symbolic analyzer for the automatic generation of analog circuit design equations," in *IEEE Int. Symp. Circuits Syst.*, Singapore, June 1991, pp. 810–813.
[19] R. Sommer, "EASY—An experimental analog design system framework," *Int. Workshop Symbolic Methods and Applicat. to Circuit Design*, Bagneux, Oct. 1991.
[20] G. DiDomenico, S. Seda, and M. Khaifa *et al.*, "BRAINS: A symbolic solver for electronic circuits," in *International Workshop on Symbolic Methods and Applications to Circuit Design*, Bagneux, Oct. 1991.
[21] S.-M. Chang, J. F. MacKay, and G. M. Wierzba, "Matrix reduction and Numerical approximation during computation techniques for symbolic analog circuit analysis," in *IEEE Int. Symp. Circuits Syst.*, San Diego, CA, May 1992, pp. 1153–1156.
[22] P. Wambacq, G. Gielen, and W. Sansen, "A cancellation free algorithm for the symbolic simulation of large analog circuits," in *IEEE Int. Symp. Circuits Syst.*, San Diego, CA, May 1992, pp. 1157–1160.
[23] F. V. Fernández, J. Martin, A. Rodríguez-Vázquez, and J. L. Huertas, "On simplification techniques for symbolic analysis of analog integrated circuits," in *IEEE Int. Symp. Circuits Syst.*, San Diego, CA, May 1992, pp. 1149–1152.
[24] S. Seda, M. Degrauwe, and W. Fichtner, "Lazy-expansion symbolic expression approximation in SYNAP," *Int. Conf. Computer-Aided Design*, Santa Clara, CA, Nov. 1992, pp. 310–317.
[25] R. R. Korfhage, *Discrete Computational Structures*. New York: Academic, 1974.
[26] C. E. Endy and P. M. Lin, " A minimization problem in systems characterized by acyclic signal flow graphs," *IEEE Trans. Circuits Syst.*, vol. CAS-28, pp. 768–780, Aug. 1981.
[27] A. V. Aho and J. D. Ullman, *Principles of Compiler Design*. Reading, MA: Addison–Wesley, 1977.
[28] J. A. Starzyk and A. Konczykowska, "Flowgraph analysis of large electronic networks," *IEEE Trans. Circuits Syst.*, vol. CAS-33, pp. 302–315, Mar. 1986.
[29] M. Hassoun and K. McCarville, "Symbolic analysis of large-scale networks using A hierarchical signal flowgraph approach," *J. Analog VLSI Signal Process.*, vol. 3, pp. 31–42, Jan. 1993.
[30] M. Hassoun and P. Atawale, "Hierarchical symbolic circuit analysis of large-scale networks on multi-processor systems," in *IEEE Int. Symp. Circuits Syst.*, Chicago, IL, May 1993, pp. 1651–1654.
[31] M. Hassoun and P.-M. Lin, "An efficient partitioning algorithm for large-scale circuits," in *IEEE Int. Symp. Circuits Syst.*, New Orleans, May 1990, pp. 2405–2408.
[32] C. Ho, A. E. Ruehli, and Brennan, "The modified nodal approach to network analysis," *IEEE Trans. Circuits Syst.*, vol. CAS-25, pp. 504–509, June 1975.
[33] B. Noble and J. Daniel, *Applied Linear Algebra*, 2nd ed. Englewood Cliffs, NJ: Prentice–Hall, 1977.
[34] G. H. Golub and C. F. van Loan, *Matrix Computations*. Oxford, England: North Oxford Academic, 1983.
[35] M. M. Hassoun, "Symbolic analysis of large-scale networks," Ph.D. thesis, School of Elec. Eng., Purdue Univ., West Lafayette, IN, Aug. 8, 1988.
[36] M. M. Hassoun and P.M. Lin, "A new network approach to symbolic simulation of large-scale networks," in *IEEE Int. Symp. Circuits Syst.*, May 1989, pp. 806–809.
[37] W.-K. Chen, *Passive and Active Filters Theory and Implementations*. New York: Wiley, 1986.



Marwan M. Hassoun (S'82–M'83) received the B.S. degree in electrical engineering with high honors from South Dakota State University, Brookings, in 1983 and the M.S. and Ph.D. degrees in electrical engineering from Purdue University, West Lafayette, IN, in 1984 and 1988, respectively.

In 1985 he was a Software Development Engineer with Hewlett–Packard Company, Santa Clara, CA, where he worked on HPSPICE. In 1988, he joined the Department of Electrical Engineering and Computer Engineering at Iowa State University, Ames, and is currently an Associate Professor. His current research interests are in symbolic analysis of VLSI circuits, magneto-resistive circuits, and analog VLSI.

Pen-Min Lin (M'64–SM'72–F'81) received the B.S.E.E. degree from National Taiwan University in 1950, the M.S.E.E. degree from North Carolina State University at Raleigh in 1956, and the Ph.D. degree in electrical engineering from Purdue University, West Lafayette, IN, in 1960.

From 1960 to 1961 he was with Bell Telephone Laboratories as a Member of the Technical Staff. Since 1961 he has been with Purdue University and is currently a Professor of Electrical Engineering Emeritus. During the academic year 1994–1995, he taught part-time at the newly established International Technological University at Santa Clara, CA.

Dr. Lin is a coauthor of the book *Computer-aided Analysis of Electronic Circuits* (Prentice–Hall, 1975), author of the book *Symbolic Network Analysis* (Elsevier, 1990), and a coauthor of the book *Linear Circuit Analysis* (Prentice–Hall, 1995).