Stable and Efficient Reduction of Large, Multiport RC Networks by Pole Analysis via Congruence Transformations

Kevin J. Kerns and Andrew T. Yang

Department of Electrical Engineering
University of Washington

Abstract — **A novel technique is presented which employs Pole Analysis via Congruence Transformations (PACT) to reduce RC networks in a well-conditioned manner. Pole analysis is shown to be more efficient than Padé approximations when the number of network ports is large, and congruence transformations preserve the passivity (and thus absolute stability) of the networks. Networks are represented by admittance matrices throughout the analysis, and this representation simplifies interfacing the reduced networks with circuit simulators as well as facilitates realization of the reduced networks using RC elements. A prototype SPICE-in, SPICE-out, network reduction CAD tool called RCFIT is detailed, and examples are presented which demonstrate the accuracy and efficiency of the PACT algorithm.**

1. INTRODUCTION

The trends in industry are to design CMOS VLSI circuits with smaller devices, higher clock speeds, lower power consumption, and more integration of analog and digital circuits; and these increase the importance of modeling layout-dependant parasitics. Resistance and capacitance of interconnect lines can delay transmitted signals. Supply line resistance and capacitance, in combination with package inductance, can lead to large variations of the supply voltage during digital switching and degrade circuit performance. In mixed-signal designs, the current injected into the substrate beneath digital devices may create significant noise in analog components through fluctuations of the local substrate voltage.

In order for designers to accurately assess on-chip layout-dependent parasitics *before fabrication*, macromodels are extracted from a layout and included in the netlist used for circuit simulation. Very often, these effects are modeled solely with lumped resistors and capacitors which form linear, multiport networks [1][2][3]. However, the networks are typically so large that simulation becomes impractical or impossible given time and memory constraints. Fortunately, most contain a relatively small number of nodes, referred to as *port nodes*, which connect the network to the rest of the circuit, and a larger number of *internal nodes*. It is possible to replace a linear network with one containing a smaller number of internal nodes while retaining the important port characteristics so that accurate simulation can be performed efficiently. The process of finding such a network is referred to as *network reduction*. It is important to ensure the absolute stability of the reduced networks so that artificial oscillations are not introduced during circuit simulations.

The most well known method used to approximate the multiport characteristics of general linear networks is Asymptotic Waveform Evaluation (AWE) [4]. In this approach, the Laplace domain moments of the port characteristics are iteratively calculated, and these are subsequently used to find poles and residues via the Padé approximation. There are several issues which make AWE a less than ideal choice for synthesizing reduced RC networks. First, the calculation of higher moments becomes ill-conditioned, so that increasing the number of moments used in the approximation does not guarantee a better fit. Heuristics have been developed which overcome this problem by repeating the moment expansion at several frequencies, but these are more computationally expensive (see [5] and references therein). Second, *asymptotic stability* of the reduced network can be maintained by dropping positive poles, but *absolute stability* is not easily ensured.

A new multiport reduction technique has been developed called "Matrix Padé via a Lanczos-type process" (MPVL) [6]. This approach avoids the ill-conditioning of AWE by using a non-symmetric, blocked version of the Lanczos algorithm to build a reduced order matrix which approximates the behavior of the original network. The basic Lanczos algorithm is discussed in detail in Section 3. Although the moments of the network are never directly calculated, the method is a Padé approximation as the reduced order matrix is formulated in such a way that moments are implicitly matched and the individual terms of the matrix are representable as poles and residues. One disadvantage of the application of MPVL to RC networks with a large number of ports is that MPVL must store two dense matrices in working memory whose dimensions are equal to the product of the number of port nodes and the number of internal nodes. It will be shown in Section 4 that storage requirements and vector computations increase as the square of the number of ports. Finally, published work on MPVL does not address the absolute stability of reduced networks.

In [7], which presents an algorithm to reduce large RC macromodels of the substrate in CMOS layouts, a derivation of multiport admittance based on partitions of the original network matrices was introduced. It was shown that passivity could be ensured by formulating the reduction as congruence transformations applied to the partitions. Here, the symmetric Lanczos process was used to form a Padé approximation of the multiport admittance characteristics in a manner as well-conditioned as MPVL, and the method has the same memory property as MPVL when a large number of ports is used.

In this paper, we present a new formulation, referred to as Pole Analysis via Congruence Transformations (PACT), which is fundamentally different from the previous methods because it is based on a pole analysis of the network and is not a Padé approximation. In addition to being as well-conditioned as MPVL and preserving passivity, our strategy addresses the prevention of error due to loss of orthogonality in the Lanczos process, while improving memory efficiency and reducing the number of vector products required when the number of ports is large. The number of internal nodes is assumed to be more important than the number of branches, as the process is designed for the reduction of large multiport 3-D RC mesh networks with strongly connected internal nodes. Internal nodes are eliminated by transforming admittance

33rd Design Automation Conference ®

DAC 96 - 06/96 Las Vegas, NV, USA

matrices representing the full network into smaller equivalent matrices which can be directly incorporated into a circuit simulator or can be used to generate SPICE-compatible RC netlists. The transformations are formulated to preserve the poles of the network between DC and a user-specified maximum frequency.

This paper details the PACT algorithm and presents a prototype CAD tool utilizing the technique. Standard mathematical notation and a review of the formulation of the multiport admittance of a general RC network are presented in Section 2. In Section 3, congruence transforms are introduced which reduce the size of an RC network using pole analysis. Section 4 provides a discussion of the memory requirement and computational complexity of the algorithm. The use of PACT in a SPICE-in, SPICE-out, RC network reduction CAD tool called RCFIT is outlined in Section 5, and several examples of reduction using RCFIT are furnished in Section 6. No proofs are provided with this manuscript, but these are presented in the journal article [8].

2. FORMULATION

This section reviews the multiport admittance formulation given in [7] which is required for Section 3. The math conventions used in this paper are adopted from Golub and Van Loan [9]. All variables are real with the exception of *i*…*n* which are integers, and *s* which is complex frequency. Under this convention, *x* is a scalar, **x** is a vector, **X** is a matrix, $\mathbf{X}^{\mathbf{T}}$ is the transpose of **X**, $\mathbf{X}^{-\mathbf{T}}$ is the inverse of the transpose of **X**, and x_{ij} is the element in the *ith* row and *j*th column of **X**. All vectors are column vectors except where noted, and the identity and zero matrices are represented as **I** and **0** respectively.

The admittance of an RC network with $m+1$ port nodes and n internal nodes can be represented by the symmetric conductance and susceptance matrices, **G** and **C**. The network has m ports because one of the port nodes is a common node. These matrices have $m+n$ rows and columns and relate nodal voltage, **x**, and injected current, **b**, in the Laplace frequency domain as

$$(\mathbf{G} + s\mathbf{C})\,\mathbf{x} = \mathbf{b}\,. \tag{1}$$

If the values of the resistors and capacitors are positive, then each diagonal entry of **G** and **C** is positive and greater than or equal to the sum of the absolute value of the off-diagonal elements in the corresponding row. These conditions are sufficient, but not necessary, to ensure that the matrices are non-negative definite. Non-negative definite symmetric matrices are ones for which all of the eigenvalues are greater than or equal to zero. As will be shown later, this property is related to the passivity of the network.

To formulate the multiport admittance, $\mathbf{Y}(s)$, of the network, a partitioning of (1) is formed by ordering the entries so that the common node is node 0, and so that the first m rows correspond to the other port nodes and the final n rows to internal nodes. Eq. (1) is rewritten as

$$\left(\underbrace{\begin{bmatrix} \mathbf{A} & \mathbf{Q}^{\mathbf{T}} \\ \mathbf{Q} & \mathbf{D} \end{bmatrix}}_{\mathbf{G}} + s \underbrace{\begin{bmatrix} \mathbf{B} & \mathbf{R}^{\mathbf{T}} \\ \mathbf{R} & \mathbf{E} \end{bmatrix}}_{s\mathbf{C}} \right) \underbrace{\begin{bmatrix} \mathbf{x}' \\ \mathbf{x}'' \end{bmatrix}}_{\mathbf{x}} = \underbrace{\begin{bmatrix} \mathbf{b}' \\ \mathbf{0} \end{bmatrix}}_{\mathbf{b}}, \tag{2}$$

and $\mathbf{x}'$ and $\mathbf{x}''$ represent the m port node voltages and the n internal node voltages respectively. The internal node partition of **b** is zero because current cannot be injected into these nodes. The $m \times m$ matrices **A** and **B** are referred to as *port matrices* since they describe the branch interconnects between the port nodes. In the same way, the $n \times n$ matrices **D** and **E** are *internal matrices*. The $n \times m$ matrices **Q** and **R** are called *connection matrices* as they describe the branches that connect internal nodes to port nodes. The matrices **A**, **B**, and **E** are symmetric and non-negative definite, and it can be show that **D** is symmetric and positive definite (non-singular) if all resistances in the network are greater than zero and if, for each internal node, there exists a DC path to a port node. The definiteness of **D** is key to the subsequent derivation of the algorithm, and is assumed to be true. Eq. (2) provides two equations with the two unknowns, $\mathbf{x}'$ and $\mathbf{x}''$. Using the definition $\mathbf{Y}(s)\mathbf{x}' = \mathbf{b}'$, and eliminating $\mathbf{x}''$ gives

$$\mathbf{Y}(s) = \mathbf{A} + s\mathbf{B} - (\mathbf{Q} + s\mathbf{R})^{\mathbf{T}} (\mathbf{D} + s\mathbf{E})^{-1} (\mathbf{Q} + s\mathbf{R})\,. \tag{3}$$

The poles of $\mathbf{Y}(s)$ occur where $(\mathbf{D}+s\mathbf{E})$ is singular, and are equal to $-\lambda^{-1}$ where λ is the solution to the general eigenvalue problem

$$\det[\mathbf{E} - \lambda\mathbf{D}] = 0\,. \tag{4}$$

Since **E** is symmetric non-negative definite, and **D** is symmetric positive definite, the poles of $\mathbf{Y}(s)$ are real and less than zero.

3. CONGRUENCE TRANSFORMS

A congruence transformation of a square matrix **W** is defined as the transformation $\mathbf{X} = \mathbf{V}^{\mathbf{T}}\mathbf{W}\mathbf{V}$. Here, **V** is referred to as the *congruence transform*, and the matrices **W** and **X** are said to be congruent. A fundamental property of square, non-singular congruence transforms is that they preserve the eigenvalues of the generalized symmetric eigenvalue problem. For example, the eigenvalues given by

$$\det\left[\mathbf{V}^{\mathbf{T}}\mathbf{E}\mathbf{V} - \lambda\mathbf{V}^{\mathbf{T}}\mathbf{D}\mathbf{V}\right] = 0 \tag{5}$$

(and therefore the poles of $\mathbf{Y}(s)$) are identical to those given by (4) if **V** is square and non-singular. The *size* of a network represented by **G** and **C** can be reduced through the application of a congruence transform with fewer columns than rows since the dimension of the resulting matrices is equal to the number of columns in the transform matrix. Although non-square transforms do not necessarily preserve the poles of $\mathbf{Y}(s)$, some can be formulated which preserve a selected set of poles. Another important property of all congruence transforms, including those which are non-square or singular, is that the definiteness and symmetry of the original matrix is preserved, and it was shown in [7] that a necessary and sufficient condition for RC networks to be passive is that the conductance and susceptance matrices representing the networks be non-negative definite. As a result, *any passive RC network which is reduced by congruence transformations remains passive.*

The reduction of a network is accomplished using two separate congruence transforms which are formulated to preserve admittance poles without matching any, but the first two, moments, and each is presented in a separate subsection. First, a square transform converts the internal conductance matrix, **D**, to the identity matrix and the connection conductance matrix, **Q**, to zero. Unwanted poles are then isolated and eliminated with a second, non-square transform.

3.1 TRANSFORMATION BASED ON CHOLESKY FACTORIZATION

The first congruence transform is used to convert the internal conductance matrix, **D**, into the identity matrix and to eliminate the connection conductance matrix, **Q**. The transform based on the Cholesky factorization $\mathbf{L}\mathbf{L}^{\mathbf{T}} = \mathbf{D}$ (**L** is a lower triangular matrix) is

$$\mathbf{V} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ -\mathbf{X} & \mathbf{L}^{-\mathbf{T}} \end{bmatrix}. \tag{6}$$

The corresponding transformation of the network is

$$\mathbf{G}' = \mathbf{V}^{\mathbf{T}}\mathbf{G}\mathbf{V} = \begin{bmatrix} \mathbf{A} - \mathbf{Q}^{\mathbf{T}}\mathbf{X} & \mathbf{0} \\ \mathbf{0} & \mathbf{L}^{-1}\mathbf{D}\mathbf{L}^{-\mathbf{T}} \end{bmatrix} = \begin{bmatrix} \mathbf{A}' & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \tag{7}$$

and

$$\mathbf{C}' = \mathbf{V}^{\mathbf{T}}\mathbf{C}\mathbf{V} = \begin{bmatrix} \mathbf{B} - \mathbf{P}^{\mathbf{T}}\mathbf{X} - \mathbf{X}^{\mathbf{T}}\mathbf{R} & \mathbf{P}^{\mathbf{T}}\mathbf{L}^{-\mathbf{T}} \\ \mathbf{L}^{-1}\mathbf{P} & \mathbf{L}^{-1}\mathbf{E}\mathbf{L}^{-\mathbf{T}} \end{bmatrix} = \begin{bmatrix} \mathbf{B}' & \mathbf{R}'^{\mathbf{T}} \\ \mathbf{R}' & \mathbf{E}' \end{bmatrix}, \quad (8)$$

where $\mathbf{X} = \mathbf{D}^{-1}\mathbf{Q}$, $\mathbf{W} = \mathbf{E}\mathbf{X}$, and $\mathbf{P} = \mathbf{R} - \mathbf{W}$ are intermediate variables. Eq. (3) is rewritten using the partitions of $\mathbf{G}'$ and $\mathbf{C}'$ as

$$\mathbf{Y}(s) = \mathbf{A}' + s\mathbf{B}' - s^2\mathbf{R}'^{\mathbf{T}}(\mathbf{I} + s\mathbf{E}')^{-1}\mathbf{R}', \quad (9)$$

and a straightforward, but tedious, substitution of variables shows that $\mathbf{Y}(s)$ in (9) is identical to that in (3).

3.2 Transformation Based on Pole Analysis

Since the poles of the network occur where $(\mathbf{I}+s\mathbf{E}')$ is singular, they can be isolated by diagonalizing $\mathbf{E}'$, and this is done using the symmetric eigendecomposition, $\mathbf{E}' = \mathbf{U}\Lambda\mathbf{U}^{\mathbf{T}}$. The diagonal matrix, Λ, contains the eigenvalues of $\mathbf{E}'$, and $\mathbf{U}$ is the square matrix whose columns are the corresponding eigenvectors. $\mathbf{U}$ is orthonormal meaning $\mathbf{U}^{\mathbf{T}}\mathbf{U} = \mathbf{I}$. The eigenvectors diagonalize $\mathbf{E}'$ in the transformations

$$\mathbf{G}'' = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{U}^{\mathbf{T}} \end{bmatrix} \begin{bmatrix} \mathbf{A}' & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{U} \end{bmatrix} = \begin{bmatrix} \mathbf{A}' & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \quad (10)$$

and

$$\mathbf{C}'' = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{U}^{\mathbf{T}} \end{bmatrix} \begin{bmatrix} \mathbf{B}' & \mathbf{R}'^{\mathbf{T}} \\ \mathbf{R}' & \mathbf{E}' \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{U} \end{bmatrix} = \begin{bmatrix} \mathbf{B}' & \mathbf{R}''^{\mathbf{T}} \\ \mathbf{R}'' & \mathbf{E}'' \end{bmatrix}, \quad (11)$$

so that $\mathbf{E}'' = \mathbf{U}^{\mathbf{T}}\mathbf{E}'\mathbf{U} = \Lambda$ is the diagonal matrix containing the eigenvalues of $\mathbf{E}'$. $\mathbf{Y}(s)$ is preserved if $\mathbf{U}$ is square and non-singular, and this is true if $\mathbf{U}$ contains the complete set of eigenvectors. Now that both internal matrices are diagonal, (9) can be written as

$$\mathbf{Y}(\mathbf{s}) = \mathbf{A}' + s\mathbf{B}' - \frac{s^2\mathbf{r}_1^{\mathbf{T}}\mathbf{r}_1}{1 + s\mathbf{e}_{11}} - \ldots - \frac{s^2\mathbf{r}_n^{\mathbf{T}}\mathbf{r}_n}{1 + s\mathbf{e}_{nn}}, \quad (12)$$

where $\mathbf{r}_i$ is the ith row of $\mathbf{R}''$ and e_{ii} is the ith diagonal of $\mathbf{E}''$. Each of the n terms associated with a pole in (12) represents the transfer function of a first order high pass filter, and the size of the network is reduced without significantly affecting the low frequency behavior by dropping those terms with pole frequencies larger than a specified cutoff frequency, $2\pi f_c = \lambda_c^{-1}$. Terms are dropped by eliminating the rows and columns of $\mathbf{G}''$ and $\mathbf{C}''$ for which the corresponding diagonal of $\mathbf{E}''$ is less than λ_c.

The Lanczos algorithm, which efficiently finds the extreme eigenvalues and eigenvectors of a large symmetric matrix [10], is ideal for this application because rows and columns containing unwanted poles can be dropped by eliminating the corresponding eigenvectors (columns) in $\mathbf{U}$ before the transformation given by (10) and (11) is performed. Only a small subset of the eigenvalues and eigenvectors of $\mathbf{E}'$ need to be calculated, namely those with eigenvalues greater than λ_c. The most important property of the algorithm is that no modification of $\mathbf{E}'$ is required so that sparsity can be maintained by avoiding a direct calculation — the solution to $\mathbf{E}'\mathbf{x}$ is found by the product of sparse matrices $\mathbf{L}^{-1}\mathbf{E}\mathbf{L}^{-\mathbf{T}}\mathbf{x}$.

The basic Lanczos algorithm is expressed by the recursion

$$\tilde{\mathbf{w}}_{j+1} = \mathbf{A}\mathbf{w}_j - \alpha_j\mathbf{w}_j - \beta_{j-1}\mathbf{w}_{j-1}, \quad (13)$$

where

$$\alpha_j = \mathbf{w}_j^{\mathbf{T}}\mathbf{A}\mathbf{w}_j \quad (14)$$

$$\beta_j = \left\|\tilde{\mathbf{w}}_{j+1}\right\|_2 \text{ (Euclidean norm)} \quad (15)$$

and

$$\mathbf{w}_{j+1} = \frac{\tilde{\mathbf{w}}_{j+1}}{\beta_j}. \quad (16)$$

To initiate the recursion, $\mathbf{w}_1$ is set to a random vector with a norm of 1, and β_0 is set to 0. The symmetric matrix to be evaluated is $\mathbf{A}$, and the vectors $\mathbf{w}_1 \ldots \mathbf{w}_k$ are called the *Lanczos vectors*. These vectors are orthonormal, meaning that $\mathbf{w}_i^{\mathbf{T}}\mathbf{w}_j = 0$ for $i \neq j$, and $\mathbf{w}_i^{\mathbf{T}}\mathbf{w}_j = 1$ for $i = j$. Only two vectors need to be in working memory at a single time, therefore the memory and the number of vector products required to calculate the next vector does not increase with the number of iterations, k. The terms from (14) and (15) are used to form the symmetric, tridiagonal matrix

$$\mathbf{T} = \begin{bmatrix} \alpha_1 & \beta_1 & & 0 \\ \beta_1 & \alpha_2 & \ddots & \\ & \ddots & \ddots & \beta_{k-1} \\ 0 & & \beta_{k-1} & \alpha_k \end{bmatrix}, \quad (17)$$

and the eigenvalues of $\mathbf{T}$, known as the *Ritz values*, approximate the eigenvalues of $\mathbf{A}$. The Ritz values generally converge first to the extreme eigenvalues of $\mathbf{A}$, and additional vectors are calculated until all of the Ritz values in a specified range have converged. The associated approximate eigenvectors of $\mathbf{A}$, known as the *Ritz vectors*, can then be found by

$$\mathbf{U} = \mathbf{W}\mathbf{Z} \quad (18)$$

where $\mathbf{Z}$ is the $k \times k$ matrix of eigenvectors of $\mathbf{T}$, the k columns of $\mathbf{W}$ are the Lanczos vectors, and the columns of $\mathbf{U}$ are the Ritz vectors corresponding to the k Ritz values. Two shortcomings of the basic algorithm are that only one eigenvalue of a group of repeated eigenvalues is found and that the rate of convergence is reduced when two or more eigenvalues are close together.

In practice, orthogonality of the Lanczos vectors is lost (i.e. $\mathbf{w}_i^{\mathbf{T}}\mathbf{w}_j \neq 0$ for $i \neq j$) after a sufficient number of iterations so that incorrect eigenvalues may be found. Variations of the algorithm deal with the loss of orthogonality in different ways. The simplest orthogonalizes each Lanczos vector with all previous vectors [11]. Although it is accurate, this method tends to be inefficient because the memory required increases linearly with k and the number of vector products required increases quadratically. At the other extreme, no extra reorthogonalization is performed, and *spurious* eigenvalues resulting from the loss of orthogonality are identified and eliminated as a post processing step [12]. A compromise algorithm, called Lanczos Algorithm with Selective Orthogonalization (LASO) [13], maintains orthogonality with *selective orthogonalization* whereby new Lanczos vectors are orthogonalized against the small set of converged Ritz vectors when a potential loss of orthogonality is detected; that is, the step

$$\tilde{\mathbf{w}}_{j+1} = \tilde{\mathbf{w}}_{j+1} - \mathbf{U}_j\left(\mathbf{U}_j^{\mathbf{T}}\tilde{\mathbf{w}}_{j+1}\right) \quad (19)$$

is added after (13) whenever the right-most term of (19) is significantly non-zero. The columns of $\mathbf{U}_j$ contain the Ritz vectors which have converged by iteration j. This method is best suited when a small subset of eigenvalues and eigenvectors is required, and, in addition, it automatically finds multiple eigenvalues. Because of these features, LASO is used by RCFIT which is discussed in Section 5.

4. Computational Complexity of the PACT Algorithm

Here, the numerical operation and memory requirements of the methods outlined in the previous sections are compared to similar methods based on the Padé approximation, namely the algorithms given in [6] and [7]. The assumption is made that the number of internal nodes in the original network, n, is larger than, but proportional to, the number of ports, m, and the number of branches connected to each node is independent of m. Requirements are analyzed for the case in which the lowest frequency pole of the

network is greater than the cutoff frequency so that no poles are retained. This analysis provides a good estimate of the memory and computational complexity when the number of low frequency poles is small.

Referring to Section 3.1, factorization of the internal conductance matrix requires $O(m^{1.0\ldots1.5})$ operations and $O(m\ldots m\log m)$ storage [14], and these are identical to those of the Padé-based methods. The left-most and right-most ranges in the operation and storage expressions of this section correspond to tree-like and mesh-like networks respectively. Calculation of $\mathbf{A}'$ and $\mathbf{B}'$ requires $O(m^2)$ memory to store the lower triangle of each matrix, and $O(m^2\ldots m^2\log m)$ operations. The requirements for $\mathbf{A}'$ and $\mathbf{B}'$ are dominant when m is very large, and, because these matrices are equal to the first two moments of $\mathbf{Y}(s)$ expanded at $s = 0$, the Padé-based methods have identical requirements for these moments.

Referring to Section 3.2, the memory required for LASO (2 Lanczos vectors of length n) is $O(m)$, and the operations required are $O(m^2\ldots m^2\log m)$ assuming the number of iterations required to find the first pole is linearly dependant on m. The Padé-based methods use the block Lanczos process with a block size of $m+1$, and the first pole is identified after the calculation of at least two blocks. In this method, $m+1$ Lanczos vectors are stored and used to orthogonalize each new vector; therefore, this method requires $O(m^2)$ memory and $O(m^3)$ operations to find the first pole. The memory requirement ($m\times n$ elements) is greater than that for the port matrices (m^2 elements) because n is typically much larger than m. As a result, the Padé-based methods are significantly less efficient when m is large and the number of low frequency poles is small.

5. RCFIT

RCFIT is a SPICE-in, SPICE-out RC network reduction CAD tool, and a flowchart is shown in Figure 1. The two columns on the right side of the figure show a graphical depiction of the conductance and susceptance matrices after each step. First, an input parser reads a SPICE netlist, and extracts all RC elements. Any node in the netlist which is connected to a resistor or capacitor as well as to a device other than a resistor or capacitor is made a port node.

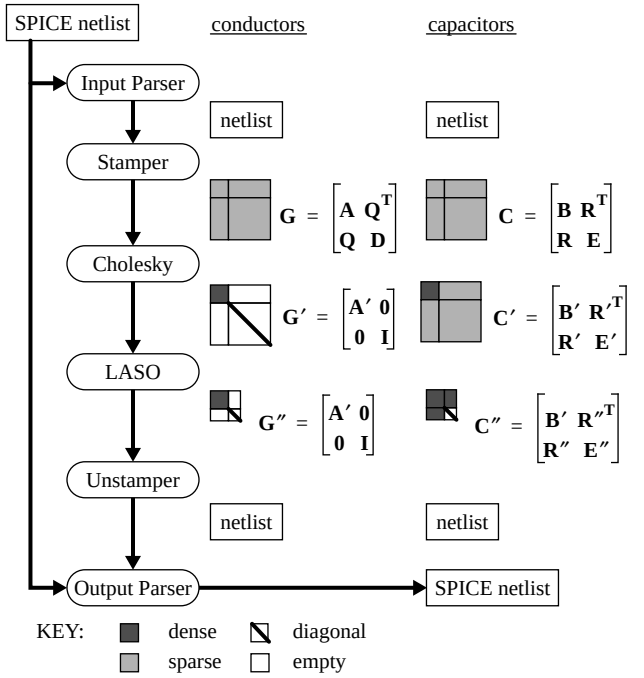

Figure 1. Flowchart of RCFIT. The two columns on the right depict the conductance and susceptance matrices after each step.

The RC netlist is loaded into sparse matrices containing the partitions of **G** and **C** in a process called stamping. The transforms given in Section 3 are then used to reduce the matrices. The transform based on Cholesky factorization, causes **A** and **B** to be dense, **Q** to be zeroed, and **D** to be diagonalized. It was shown in [7] that this sparse factorization is an order of magnitude faster than sparse LU factorization. Memory is conserved by calculating one column at a time of the intermediate matrices associated with (6…8) so that only n elements need to be stored. For the same reason, calculation of $\mathbf{R}'$ is delayed until it is needed for the transform given by (11). The second transform uses LASO to find eigenvectors for eigenvalues below a cutoff frequency which is set so that the error at the user-specified maximum frequency is bounded by the user-specified error tolerance (e.g., a 5% tolerance requires the cutoff frequency to be 3.04 times larger than the maximum frequency). The eigenvectors are used to diagonalize the internal susceptance matrix, **E**, and to reduce the number of internal nodes. Before the matrices are unstamped, sparsity is enhanced using a heuristic which drops very small off-diagonal elements while maintaining passivity. The unstamped netlists are sent to a parser that outputs a SPICE netlist which is identical to the original except the RC networks have been replaced by their reduced equivalents.

6. EXAMPLES

All examples are accomplished on a Sun SPARC-20 workstation using RCFIT and HSPICE [15]. An illustrative example is first presented which is simple enough that the original netlist and the reduced admittance matrices can be easily depicted. Figure 2 shows a large CMOS inverter which drives a second inverter over a long line with a distributed resistance and capacitance of 250 Ω and 1.35 pF respectively. The line is modeled using a 100 segment RC ladder, and the corresponding SPICE netlist is reduced by RCFIT with a specified error tolerance of 5% and maximum frequency of 5 GHz. A single pole is found at 4.7 GHz — thus, the reduced network has one internal node. The admittance is specified as

$$\mathbf{G} = \begin{bmatrix} 4 & -4 & 0 \\ -4 & 4 & 0 \\ 0 & 0 & 32 \end{bmatrix} \text{m}\mho, \text{ and } \mathbf{C} = \begin{bmatrix} 443 & 225 & -547 \\ 225 & 457 & -547 \\ -547 & -547 & 1094 \end{bmatrix} \text{fF}, \quad (20)$$

where the first two rows of the matrices correspond to the port nodes 1 and 2. In Figure 3, which shows the results of a transient simulation, the effect of the transmission line can be seen by comparing results with the line removed and with the 100 segment line in place. A two segment line, whose total resistance and capacitance are identical to those of the original line, is also included to show that (20) gives a better fit while using the same number of internal nodes.

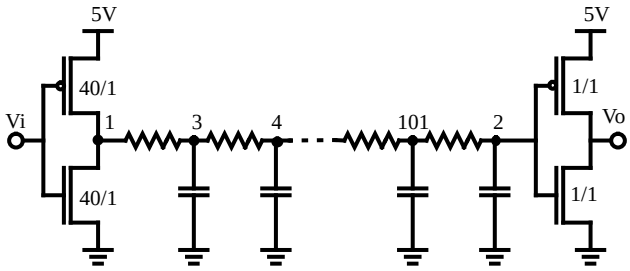

Figure 2. CMOS inverter pair separated by an RC transmission line with a distributed resistance and capacitance of 250Ω and 1.35 pF. The line is modeled with 100 lumped segments.

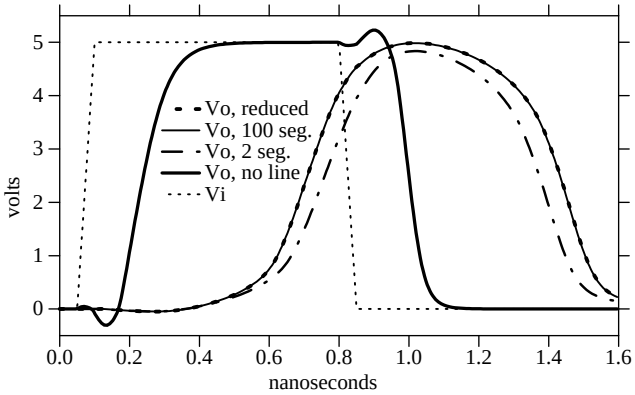

Figure 3. Effect of transmission line models on output voltage for the circuit in Figure 2.

The next example shows the reduction of the RC network model for interconnect parasitics in a CMOS 8 bit digital multiplier containing 7264 transistors. Simulations were performed for the circuit without parasitics, with the original RC network model, and with the network reduced using a tolerance of 5% and frequency of 500 MHz. The output of one of the critical paths from an HSPICE simulation is shown in Figure 4, and statistics on the individual simulations are given in Table 1. It is clear that the critical path is significantly affected by layout parasitics, and the reduced network accurately models these effects while reducing simulation time by 12%. The time and memory savings are not large in this example because the time required to analyze the (non-linear) transistors is large compared to that required to analyze the RC interconnect network.

The next example shows the reduction of a 3-D RC mesh used to simulate voltage fluctuations in the substrate which result from current injected due to switching activity in a nearby CMOS one-bit full adder. The adder contains 22 MOSFETs, and each body terminal node is a port into the substrate macromodel. Two port nodes are also used for the Vdd and Vss well and substrate contacts, and an additional one is included to monitor the substrate voltage at a point near the adder. Therefore, the total number of port nodes is 25. The total number transistors in the circuit is 28 as the three inputs to the adder are driven by separate CMOS inverters.

Table 1. Reduction and simulation statistics for the CMOS 8 bit multiplier containing 7264 transistors.

Simulation	Total Nodes	R's	C's	RCFIT Reduction		HSPICE Simulation	
				Time (sec)	Mem. (Mb)	Time (sec)	Mem. (Mb)
no RC	3650	0	23	—	—	5855.9	31.1
with RC	13770	10120	10143	—	—	7563.8	37.2
reduced RC	5674	2031	4914	4.1	0.6	6621.5	31.1

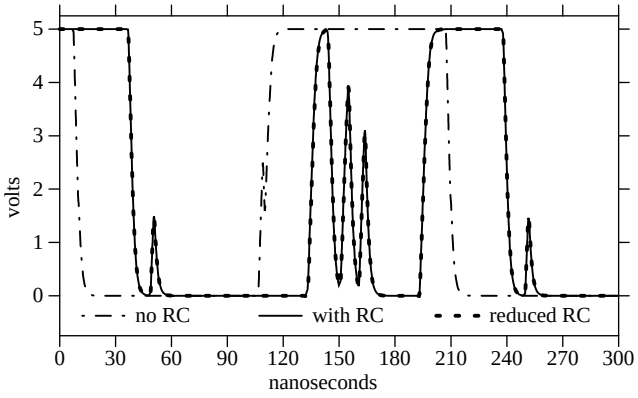

Figure 4. Effect of RC interconnect parasitics on a critical path of the 8 bit digital multiplier.

Table 2. Reduction and simulation of the substrate mesh with 25 port nodes for the one-bit full adder. The AC sweep was performed for 81 sampled frequencies.

Maximum Frequency	Total Nodes	R's	C's	RCFIT Reduction			HSPICE AC	
				Poles	Time (sec)	Mem. (Mb)	Time (sec)	Mem. (Mb)
(original)	1525	4970	253	—	—	—	1841.5	47.6
3 GHz	31	125	408	6	6.2	1.4	1.5	0.4
1 GHz	26	120	278	1	6.2	1.4	1.2	0.4
300 MHz	25	119	227	0	5.9	1.4	1.1	0.3

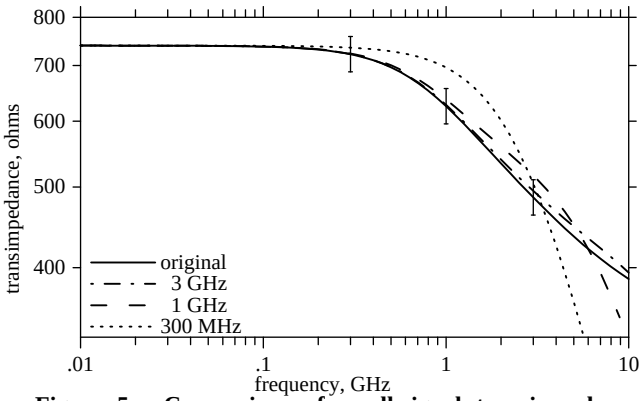

Figure 5. Comparison of small-signal transimpedance between two port nodes for the reduced meshes given in Table 2. The error bars indicate 5% error at the specified maximum frequencies.

The statistics on the reduction of the mesh using several frequencies and a 5% tolerance are shown in Table 2. The last columns of the table show the time and memory required to perform the frequency domain analysis presented in Figure 5 which depicts the magnitude of the small-signal transimpedance between the monitor port and one of the NMOS ports of the network. The error bars, which indicate 5% error relative to the transimpedance of the original network at the three frequencies, show that each reduced network is accurate to within 5% for frequencies below the specified maximum.

Figure 6 shows the results of an HSPICE transient simulation

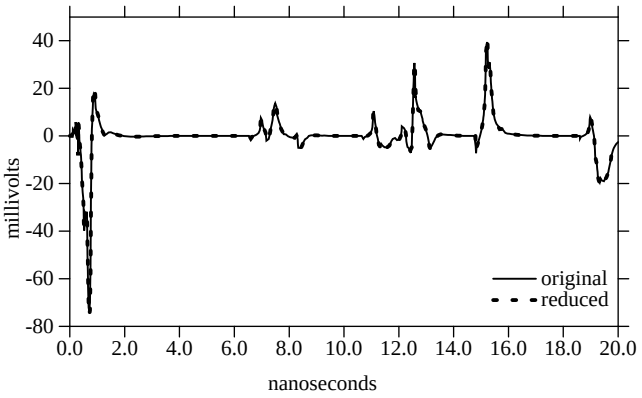

Figure 6. Simulation of voltage fluctuations in the substrate which result from activity in the one-bit full adder.

Table 3. Reduction and transient simulation statistics of the one-bit full adder circuit.

Substrate Network	Total Nodes	Total RC's	RCFIT Reduction		HSPICE Simulation	
			Time (sec)	Mem. (Mb)	Time (sec)	Mem. (Mb)
original	1540	5256	—	—	12511.6	44.9
reduced, 1GHz	41	431	6.2	1.4	40.0	0.4

of the one-bit full adder circuit. Substrate voltage fluctuations are compared for simulations with the original substrate mesh and with the reduced mesh using the 1 GHz frequency. As can be seen, the reduced network gives a very good approximation to the behavior of original network. Table 3 provides statistics on the reduction and simulation of the network, and shows that use of the reduced network speeds the simulation over 300 times while reducing the memory required by two orders of magnitude.

Comparison of the memory requirements for the examples given in Tables 1 and 3 illustrates an important difference. The HSPICE simulation of the circuit using the full RC network in the former example, which has 7264 MOSFETs and 20263 RC elements, requires 37.2 Mbytes memory, while that for the later example, which has only 28 MOSFETs and 5223 RC elements, requires 44.9 Mbytes! The discrepancy in the relative memory requirements results from differences in the circuit topologies. In the prior example, both transistors and RC elements form a nearly tree-like structure so that the matrix used by HSPICE to represent the circuit, is very sparse, and the factorization of that matrix is nearly as sparse. In the subsequent example, the transistors are tree-like, but the RC elements form a strongly connected 3-D mesh. As a result, the factorization of the circuit matrix is much less sparse and more memory is required to contain it. This example elucidates the importance of minimizing the number of *internal nodes*, and not branches, during the reduction of mesh networks.

The final example shows the reduction of a very large 3-D RC mesh network which models the substrate beneath a circuit containing 467 CMOS transistors. As with the previous example, the body terminal of each transistor forms a port of the substrate network, and two additional ports are included for the substrate and well contacts. Table 4 shows statistics on the reduction of the network using a 500 MHz frequency and 10% tolerance. Of the 25.8 Mbytes total memory required, 19.5 Mbytes is used to contain the Cholesky factorization of **D**. In contrast to the remaining 6.3 Mbytes used by RCFIT, the Padé-based methods require $469 \times 19877 \times 8 = 71.1$ Mbytes for the Lanczos vectors alone, and MPVL requires two of these blocks. The original network is too large to simulate with HSPICE, so no comparisons can be made regarding the behaviors of the reduced and full networks.

Conclusions

A new algorithm, called Pole Analysis via Congruence Transformations (PACT), has been presented which is designed to reduce large multiport RC networks. Because it is based on pole analysis, and not on the Padé approximation, it is fundamentally different from AWE and MPVL. The strategy employs congruence transforms which reduce the size of the network admittance matrices in a well-conditioned manner while preserving passivity by eliminating unwanted poles, and it was shown to be more time and memory efficient than the Padé-based methods when the number of ports is large. Additionally, a prototype network reduction CAD tool, named RCFIT, was introduced, and its performance was demonstrated in several examples which show that reduction of RC networks can significantly decrease the time and memory required for circuit simulations without introducing significant error.

Table 4. Reduction of the large 3-D mesh RC network.

network	Port Nodes	Internal Nodes	R's	C's	Time (sec)	Mem. (Mb)
original	469	19877	65809	3683	–	–
reduced, 500 Mhz	469	10	14221	46427	1792.6	25.8

References

[1] H. B. Bakoglu, *Circuits, Interconnects, and Packaging for VLSI*. Reading, MA: Addison-Wesley Pub. Co., 1990.

[2] I. L. Wemple and A. T. Yang, "Mixed-signal switching noise analysis using Voronoi-tessellated substrate macromodels," *Proceedings of the 32nd ACM/IEEE Design Automation Conference*, pp. 439-444, 1995.

[3] S.-S. Lee and D. J. Allstot, "Electrothermal simulation of integrated circuits," *IEEE J. Solid-State Circuits*, vol. 28, pp. 1283-1293, December 1993.

[4] L. T. Pillage and R. A. Rohrer, "Asymptotic waveform evaluation for timing analysis," *IEEE Trans. Computer-Aided Design*, vol. 9, pp. 352-366, April 1990.

[5] V. Raghavan, R. A. Rohrer, L. T. Pillage, J. Y. Lee, J. E. Bracken, and M. M. Alaybeyi, "AWE-Inspired," *Proceedings of the IEEE 1993 Custom Integrated Circuits Conference*, pp. 18.1.3-18.1.8, 1993.

[6] P. Feldmann and R. W. Freund, "Reduced-order modeling of large linear subcircuits via a block Lanczos algorithm," *Proceedings of the 32nd ACM/IEEE Design Automation Conference*, pp. 474-479, 1995.

[7] K. J. Kerns, I. L. Wemple, and A. T. Yang, "Stable and efficient reduction of substrate model networks using congruence transforms," *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pp. 207-214, November 1995.

[8] K. J. Kerns and A. T. Yang, "Stable and efficient reduction of large, multiport RC networks by pole analysis via congruence transformations," submitted to *IEEE Trans. Computer-Aided Design*.

[9] G. H. Golub and C. F. Van Loan, *Matrix Computations, Second Edition*. Baltimore: Johns Hopkins University Press, 1993.

[10] C. Lanczos, "An iteration method for the solution of the eigenvalue problem of linear differential and integral operators," *Journal of Research of the National Bureau of Standards*, vol. 45, pp. 255-282, October 1950.

[11] C. C. Paige, "Practical use of the symmetric Lanczos process with reorthogonalization methods," *BIT*, vol. 10, pp. 183-195, 1970.

[12] J. Cullum and R. A. Willoughby, "Lanczos and the computation in specified intervals of the spectrum of large, sparse real symmetric matrices," in *Sparse Matrix Proceedings*, (1978) ed. I.S. Duff and G. W. Stewart, Philadelphia: SIAM Publications, 1979.

[13] B. N. Parlett and D. S. Scott, "The Lanczos algorithm with selective orthogonalization," *Mathematics of Computation*, vol. 33, pp. 217-238, January 1979.

[14] A. George and J.W.-H. Liu, *Computer Solution of Large Sparse Positive Definite Systems*. Englewood Cliffs, NJ: Prentice-Hall, Inc., 1981.

[15] HSPICE, Meta-Software, Inc., Version H95.1, Campbell, CA, 1995.