Available at
www.ElsevierComputerScience.com
POWERED BY SCIENCE DIRECT®
Parallel Computing 29 (2003) 1219–1235

PARALLEL COMPUTING

www.elsevier.com/locate/parco

Parallel iterative methods for dense linear systems in inductance extraction

Hemant Mahawar, Vivek Sarin *

Department of Computer Science, Texas A&M University, 410 B, H.R. Bright Building, College Station, TX 77843-3112, USA

Received 15 January 2003; received in revised form 1 June 2003; accepted 7 June 2003

Abstract

Accurate estimation of the inductive coupling between interconnect segments of a VLSI circuit is critical to the design of high-end microprocessors. This paper presents a class of parallel iterative methods for solving the linear systems of equations that arise in the inductance extraction process. The coefficient matrices are made up of dense and sparse submatrices where the dense structure is due to the inductive coupling between current filaments and the sparse structure is due to Kirchoff's constraints on current. By using a solenoidal basis technique to represent current, the problem is transformed to an unconstrained one that is subsequently solved by an iterative method. A dense preconditioner resembling the inductive coupling matrix is used to increase the rate of convergence of the iterative method. Multipole-based hierarchical approximations are used to compute products with the dense coefficient matrix as well as the preconditioner. A parallel formulation of the preconditoned iterative solver is outlined along with parallelization schemes for the hierarchical approximations. A variety of experiments is presented to show the parallel efficiency of the algorithms on shared-memory multiprocessors.

Keywords: Parallel computing; Preconditioning; Iterative methods; Parasitic extraction

1. Introduction

A key component in the design of high-end microprocessors is the estimation of signal delay of the VLSI circuit. The signal delay depends on several factors including

* Corresponding author.
E-mail address: sarin@cs.tamu.edu (V. Sarin).

0167-8191/$ - see front matter © 2003 Published by Elsevier B.V.
doi:10.1016/S0167-8191(03)00100-5

parasitic resistance, capacitance, and inductance due to the on-chip interconnect. At higher frequencies, it is critical to estimate the effect of the inductive coupling between the interconnect segments quickly and accurately.

This paper presents a class of parallel algorithms for solving the linear systems of equations that arise in inductance extraction of VLSI circuits. To obtain the parasitic impedance at a particular frequency, the interconnect segments are discretized by a uniform mesh of filaments. Each filament carries an unknown amount of current which depends on the potential difference between the end nodes of the filament. The net flow of current into any node must be zero. These conditions give rise to linear systems with large coefficient matrices that have both sparse and dense submatrices. The sparse submatrices represent the constraints on current flow at each node. The dense submatrices represent the potential drop across each filament due to the inductive effect of current in all other filaments as well as due to its own resistance.

For accurate impedance estimation, discretizations with millions of filaments may be necessary. Since the computation and storage of the associated dense submatrix is not feasible, one must use iterative methods that rely on the ability to compute matrix–vector products with the coefficient matrix. The nature of the inductive coupling between filaments permits use of hierarchical multipole-based techniques such as the fast multipole method (FMM) [3,8] to compute approximations to the matrix–vector products. The non-availability of the coefficient matrix makes it difficult to construct preconditioners that can be used to increase the rate of convergence of the iterative methods. The constraints imposed on the flow of current leads to an indefinite coefficient matrix, further complicating efforts to design robust preconditioners.

The approach presented in the paper uses a solenoidal basis method to restrict the current to the subspace where Kirchoff's current law is satisfied, and solves the resulting reduced system by a preconditioned iterative method. The preconditioners are constructed from the inductive coupling of solenoidal functions of the basis, and the preconditioning step is implemented using hierarchical multipole-based approximations. The description of the inductance problem and the solution methodology in Sections 2 and 3 follows the presentation in [7]. Section 4 describes a parallelization scheme for the hierarchical approximation algorithms used to compute matrix–vector products with the dense coefficient submatrix as well as the preconditioner. Benchmark experiments presented in Section 5 demonstrate that the algorithm can achieve high parallel efficiency on shared-memory multiprocessors. Concluding remarks are presented in Section 6.

2. Inductance extraction problem

Given a set of s conductors, the inductance extraction problem consists of finding an $s \times s$ matrix that summarizes the impedance among the conductors. The (k, l) element of this impedance matrix is equal to the potential difference generated across the kth conductor in response to a unit current source applied to the lth conductor. An entire column of the matrix can be computed by solving a linear system of equa-

tions. The impedance matrix is determined by solving s instances of the same linear system with different right-hand sides.

Each conductor is discretized by a two-dimensional uniform mesh of filaments that carry unknown current (see, e.g., Fig. 1). For a problem with n filaments, $f_1, f_2, \ldots, f_n$, one obtains the following linear system:

$$[\mathbf{R} + j\omega\mathbf{L}]\mathbf{I}_\mathrm{f} = \mathbf{V}_\mathrm{f}, \tag{1}$$

where $\mathbf{R}$ is an $n \times n$ diagonal matrix of filament resistances, $\mathbf{I}_\mathrm{f}$ is the vector of unknown filament currents, $\mathbf{V}_\mathrm{f}$ is the vector of potential differences between the ends of each filament, ω is the frequency, and $j = \sqrt{-1}$. The kth diagonal element of $\mathbf{R}$ is assigned the value $\rho l_k/a_k$, where ρ is the resistivity, l_k is the length of the kth filament, and a_k is the cross-sectional area of the kth filament. The elements of the inductance matrix $\mathbf{L}$ are given by

$$\mathbf{L}_{kl} = \frac{\mu}{4\pi}\frac{1}{a_k a_l}\int_{r_k \in V_k}\int_{r_l \in V_l}\frac{\mathbf{u}_k \cdot \mathbf{u}_l}{\|\mathbf{r}_k - \mathbf{r}_l\|_2}\,\mathrm{d}V_k\,\mathrm{d}V_l, \tag{2}$$

where $\mathbf{u}_k$ denotes the unit vector along the kth filament, $\mathbf{r}$ denotes a three-dimensional position vector, and μ is the magnetic permeability. The linear system (1) results from the discretization of the equation for current density $\mathbf{J}$ at a point $\mathbf{r}$:

$$\rho\mathbf{J}(\mathbf{r}) + j\omega\int_V \frac{\mu}{4\pi}\frac{\mathbf{J}(\mathbf{r}')}{\|\mathbf{r} - \mathbf{r}'\|_2}\,\mathrm{d}V' = -\nabla\phi(\mathbf{r}), \tag{3}$$

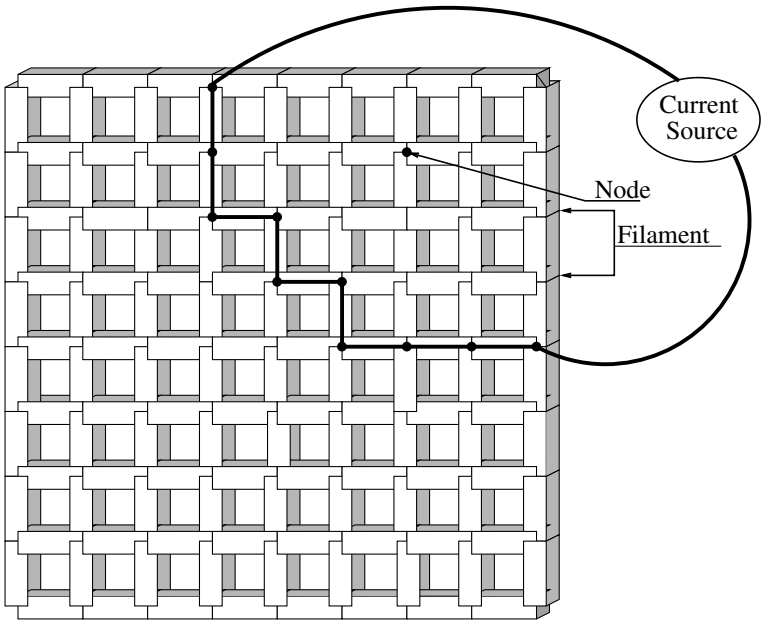

Fig. 1. Discretization of a ground plane conductor with a two-dimensional uniform mesh of filaments. The bold line indicates a path for current that satisfies boundary conditions.

where $\phi(\mathbf{r})$ is the potential at $\mathbf{r}$. The volume of conductors is denoted by V and incremental volume with respect to $\mathbf{r}'$ is denoted by $\mathrm{d}V'$. The expression for $\mathbf{L}_{kl}$ in (2) is obtained by assuming: (a) current density is constant within each filament, and (b) current flows along the length of a filament. The reader is referred to [5] for further details.

The constraints imposed on current by Kirchoff's law can be expressed as

$$\mathbf{B}^{\mathrm{T}}\mathbf{I}_{\mathrm{f}} = \mathbf{I}_{\mathrm{s}}, \tag{4}$$

where $\mathbf{B}^{\mathrm{T}}$ is the branch index matrix and $\mathbf{I}_{\mathrm{s}}$ is the external source current. The branch index matrix for a mesh of m nodes and n filaments is an $m \times n$ matrix whose (k, l) element has the value -1 if the lth filament originates at node k, 1 if it terminates at k, and zero otherwise. It turns out that the potential difference across the filaments can be expressed in terms of the node potential as follows:

$$\mathbf{V}_{\mathrm{f}} = \mathbf{B}\mathbf{V}_{\mathrm{n}}, \tag{5}$$

where $\mathbf{V}_{\mathrm{n}}$ is a vector of node potentials defined with respect to the ground.

To compute the unknown filament currents and node potentials, Eqs. (1), (4) and (5) are solved simultaneously in the form of the following linear system:

$$\begin{bmatrix} \mathbf{R} + j\omega\mathbf{L} & -\mathbf{B} \\ \mathbf{B}^{\mathrm{T}} & 0 \end{bmatrix} \begin{bmatrix} \mathbf{I}_{\mathrm{f}} \\ \mathbf{V}_{\mathrm{n}} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{I}_{\mathrm{s}} \end{bmatrix}. \tag{6}$$

A particular current flow, say $\mathbf{I}_{\mathrm{p}}$, that satisfies the boundary conditions can be found by assigning fixed current to filaments along an arbitrary path between the input and output nodes (see, e.g., Fig. 1). By subtracting $\mathbf{I}_{\mathrm{p}}$ from the unknown current, one arrives at an equivalent system with a modified right-hand side:

$$\begin{bmatrix} \mathbf{R} + j\omega\mathbf{L} & -\mathbf{B} \\ \mathbf{B}^{\mathrm{T}} & 0 \end{bmatrix} \begin{bmatrix} \mathbf{I} \\ \mathbf{V}_{\mathrm{n}} \end{bmatrix} = \begin{bmatrix} \mathbf{F} \\ \mathbf{0} \end{bmatrix}, \tag{7}$$

where $\mathbf{I}$ has been used to represent the difference between the unknown current $\mathbf{I}_{\mathrm{f}}$ in (6) and $\mathbf{I}_{\mathrm{p}}$. The main difference between (6) and (7) is that the boundary conditions are specified for current in the first system whereas they are specified for potential in the second system. This paper is concerned with the solution of the linear system in (7).

Removal of the current unknowns from (7) by a single block-step of Gaussian elimination process leads to a smaller system:

$$\mathbf{B}^{\mathrm{T}}[\mathbf{R} + j\omega\mathbf{L}]^{-1}\mathbf{B}\mathbf{V}_{\mathrm{n}} = -\mathbf{B}^{\mathrm{T}}[\mathbf{R} + j\omega\mathbf{L}]^{-1}\mathbf{F}.$$

To solve the above system, one must use an iterative solver in which the product with the coefficient matrix, $y = \mathbf{B}^{\mathrm{T}}[\mathbf{R} + j\omega\mathbf{L}]^{-1}\mathbf{B}x$, is computed as a sequence of three steps: (i) the product $u = \mathbf{B}x$, (ii) solution of the system $[\mathbf{R} + j\omega\mathbf{L}]v = u$, and (iii) the product $y = \mathbf{B}^{\mathrm{T}}v$. Since the second step requires an inner iterative solver, the outer iterations tend to be very expensive. Moreover, the structure of the coefficient matrix makes it very difficult to precondition the linear system.

3. A preconditioned iterative solver

An alternate approach to solving the linear system in (7) uses a basis for the subspace of current vectors that satisfy the constraint $\mathbf{B}^{\mathrm{T}}\mathbf{I} = \mathbf{0}$. For instance, given a basis $\mathbf{P}$ for the null space of $\mathbf{B}^{\mathrm{T}}$, the current vector $\mathbf{I} = \mathbf{P}x$ satisfies the constraint for arbitrary x. By restricting the current to the null space of $\mathbf{B}^{\mathrm{T}}$, the linear system in (7) is transformed to the following system:

$$[\mathbf{R} + j\omega\mathbf{L}]\mathbf{P}x - \mathbf{B}\mathbf{V}_{\mathrm{n}} = \mathbf{F}.$$

The unknown vector $\mathbf{V}_{\mathrm{n}}$ can be eliminated by multiplying the above equation with $\mathbf{P}^{\mathrm{T}}$, resulting in the reduced system

$$\mathbf{P}^{\mathrm{T}}[\mathbf{R} + j\omega\mathbf{L}]\mathbf{P}x = \mathbf{P}^{\mathrm{T}}\mathbf{F} \tag{8}$$

which is of order $(n - m + 1)$. This reduced system can be solved for x by an iterative method such as GMRES [9]. Current is given as $\mathbf{I} = \mathbf{P}x$, and potential difference across each filament is computed as follows

$$\mathbf{V}_{\mathrm{f}} = [\mathbf{R} + j\omega\mathbf{L}]\mathbf{P}x - \mathbf{F}.$$

The potential difference between two nodes is computed by adding the potential difference across filaments on any path connecting the nodes. This allows computation of impedance between the end points of a conductor.

There are several ways to compute a basis for the null space of a matrix. An algebraic approach such as the QR factorization of $\mathbf{B}$ cannot be used to compute $\mathbf{P}$ due to the prohibitive cost of computation and storage of a large dense matrix. To construct a sparse basis, observe that a current flow of fixed magnitude along any closed path in the mesh satisfies the constraints. We define a *local* solenoidal flow in the kth mesh cell to be a unit current flow in the counterclockwise direction in the four filaments that form the boundary of the cell (see, e.g., Fig. 2). Each such flow can be represented as a current vector with four non-zero elements that correspond to the directional flow of current in the four filaments. Each of the $(n - m + 1)$ cells of the mesh contribute a column vector to the null space basis $\mathbf{P}$. The local nature of these mesh currents leads to efficient computation and storage schemes for $\mathbf{P}$. Details of this scheme are presented in [7].

The reduced system (8) can be preconditioned effectively by the following matrix [7]:

$$\mathscr{M}^{-1} = \widetilde{\mathbf{L}}[\widetilde{\mathbf{R}} + j\omega\widetilde{\mathbf{L}}]^{-1}\widetilde{\mathbf{L}}, \tag{9}$$

where $\widetilde{\mathbf{R}}$ is a diagonal matrix of resistance as seen by the local mesh currents. The elements of the matrix $\widetilde{\mathbf{L}}$ are given as

$$\widetilde{\mathbf{L}}_{kl} = \frac{\mu}{4\pi}\frac{1}{a_k a_l}\int_{r_k \in V_k}\int_{r_l \in V_l}\frac{1}{\|\mathbf{r}_k - \mathbf{r}_l\|_2}\,\mathrm{d}V_k\,\mathrm{d}V_l. \tag{10}$$

The value of the element $\widetilde{\mathbf{L}}_{kl}$ equals the mutual inductance between parallel filaments placed at the centers of loop k and l. At each iteration, the preconditioning step

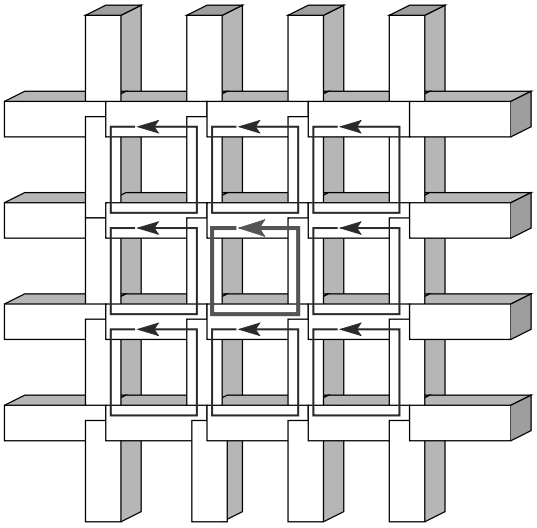

Fig. 2. Local solenoidal flows in a section of a uniform mesh.

consists of the matrix–vector product $z = \mathscr{M}^{-1} r$ which can be computed in three steps:

$$u = \widetilde{\mathbf{L}} r, \quad v = [\widetilde{\mathbf{R}} + j\omega\widetilde{\mathbf{L}}]^{-1} u, \quad z = \widetilde{\mathbf{L}} v.$$

The matrix–vector product in the first and third steps uses approximate hierarchical techniques identical to those used for $\mathbf{L}$. The second step is implemented via an inner iterative method which is used to solve the system $[\widetilde{\mathbf{R}} + j\omega\widetilde{\mathbf{L}}]v = u$. At low and high frequencies, one can use the following approximations to the preconditioner without any significant change in the rate of convergence:

$$\mathscr{M}_{\text{low}}^{-1} = \widetilde{\mathbf{L}}\widetilde{\mathbf{R}}^{-1}\widetilde{\mathbf{L}}, \quad \mathscr{M}_{\text{high}}^{-1} = -j\omega^{-1}\widetilde{\mathbf{L}}.$$

In each case, the preconditioning step is relatively cheap since it does not involve an inner solve. For intermediate frequencies, however, one must use the preconditioner in (9).

These preconditioners have shown near-optimal performance on several benchmark problems [7]. The rate of convergence of the preconditioned GMRES method is weakly dependent on the frequency ω and the discretization mesh width h. For example, Table 1 shows that the number of iterations required by right preconditioned GMRES changes only slightly with ω and h. The growth in iterations is more pronounced for 1 GHz due to the use of the high frequency preconditioner $\mathscr{M}_{\text{high}}^{-1}$ which is significantly cheaper than the one given in (9). In contrast, the number of iterations grows linearly with the mesh size when no preconditioning is used with a 129×129 mesh requiring 566 iterations. A tolerance of 10^{-3} was used for the relative residual norm. The FMM algorithm with multipole degree 1 was used to compute matrix–vector products with $\mathbf{L}$ and $\widetilde{\mathbf{L}}$.

Table 1
The number of preconditioned GMRES iterations to compute the self-impedance of the ground plane conductor problem

Mesh size	Filament length (cm)	Frequency			
		1 GHz	10 GHz	100 GHz	1 THz
65×65	2^{-6}	6	6	5	5
129×129	2^{-7}	7	6	6	6
257×257	2^{-8}	9	7	7	6
513×513	2^{-9}	12	8	7	7

A related mesh current based approach called *FastHenry* has been proposed in [5] where the preconditioners were derived from incomplete factorization of a sparse approximation to the dense reduced system. The advantages of our preconditioned iterative solver over *FastHenry* are discussed in [7].

3.1. Computing dense matrix–vector products

The most computationally intensive steps in the algorithm are the matrix–vector products with the reduced system matrix $\mathbf{P}^{\mathrm{T}}[\mathbf{R}+j\omega\mathbf{L}]\mathbf{P}$ and the preconditioner matrix $\mathscr{M}^{-1}$. A matrix–vector product with the reduced system is computed as a sequence of three products:

$$u=\mathbf{P}x,\quad v=[\mathbf{R}+j\omega\mathbf{L}]u,\quad y=\mathbf{P}^{\mathrm{T}}v.$$

The cost of multiplying a vector with the dense matrix $\mathbf{L}$ is significantly greater than multiplication with $\mathbf{P}$ or $\mathbf{P}^{\mathrm{T}}$. Since the matrix $\widetilde{\mathbf{L}}$ used in the preconditioning step is similar to $\mathbf{L}$, it is worthwhile to use fast methods to compute the matrix–vector products with $\mathbf{L}$.

The cost of computing an accurate matrix–vector product with an $n\times n$ dense matrix is $\mathrm{O}(n^2)$ operations. The nature of the elements in $\mathbf{L}$ allows use of fast hierarchical algorithms in which reduction in computational complexity is obtained at the expense of accuracy. In particular, one can exploit the rapid decay of the kernel in (2) with distance to compute approximate matrix–vector products in $\mathrm{O}(n\log n)$ or $\mathrm{O}(n)$ operations. A number of such techniques have been developed including the well known FMM [3,8], the Barnes–Hut [2] method, and Appel's algorithm [1]. For the inductance extraction problem, these algorithms may use a truncated series to approximate the effect of a cluster of filament currents on other clusters that are well-separated. The method of Barnes and Hut relies only on filament–cluster interactions to achieve an $\mathrm{O}(n\log n)$ computational bound whereas the FMM uses both filament–cluster and cluster–cluster interactions to achieve an $\mathrm{O}(n)$ bound for uniform filament distributions. In each case, reduction in computational complexity is associated with decrease in accuracy of the matrix–vector product.

The accuracy of FMM can be improved by increasing the multipole degree d which determines the number of terms used in the approximation. Even though the growth in computational complexity is proportional to d^4 [3], it is worthwhile

to choose a larger value of d to offset the error introduced by the presence of differencing operators $\mathbf{P}$ and $\mathbf{P}^{\mathrm{T}}$ in the reduced system. In this paper, we use the FMM algorithm to compute approximate matrix–vector products with both $\mathbf{L}$ and $\widetilde{\mathbf{L}}$.

4. Parallelism

To develop efficient parallel formulations of the iterative solver, it is necessary to understand the structure of the dense matrix–vector products. The product with $\mathbf{L}$ is viewed as the calculation of the potential difference across each filament due to the inductive effect of current in all other filaments. The integral in (2) is approximated by a weighted sum over a set of discrete points within each filament. The mid-point of a filament is sufficient for calculating the mutual inductance between all pairs of filaments except those in close proximity. Additional points may be used for filaments that are close to each other. Self-inductance needs to be computed using special formulas [4]. The primary goal of FMM is to reduce the complexity of computing the mutual inductance between filaments that are not in close proximity.

An oct-tree is used to compute a hierarchical spatial decomposition of the mid-points of the filaments. The root of the tree represents a cubical domain containing all the points. Eight children nodes are created by partitioning the domain into eight equal subcubes. The points are also partitioned among the subdomains. A recursive strategy yields an oct-tree with a hierarchical spatial ordering of the points. Since the oct-tree stores non-empty cubes only, this scheme yields non-uniform oct-trees for unstructured point distributions.

A node in the tree represents a subdomain and its associated oct-tree. For each node, FMM computes a set of multipole coefficients that can be used to calculate the inductive effect of the points within the subdomain at an observation point outside the subdomain. The observation point must be outside a neighborhood that contains the sphere enclosing the subdomain. For simplicity, a larger cube can be chosen as the neighborhood of a subdomain. The computational complexity of this step is $O(d^4 n)$ for a problem with n points (see, e.g., [3] for additional details). At any point, the net inductive effect due to far-off points can be determined from the multipole coefficients of only $O(\log n)$ nodes. To reduce the complexity further down to $O(n)$, FMM computes a set of local coefficients for each node that can be used to calculate the inductive effect of all the points outside its neighborhood. These coefficients are computed from the multipole coefficients of nodes that are adjacent to the leaf's neighborhood as well as from local coefficients of the leaf's parent.

The inductive effect at any point in the leaf node is calculated as a sum of two values: a *far-field* due to the points outside the leaf's neighborhood and a *near-field* due to the points that lie within the neighborhood. The far-field is computed using the local coefficients at the leaf node whereas the near-field is found by direct computation.

The domain is partitioned into non-overlapping subdomains by pruning the oct-tree at a particular level k (see, e.g., Fig. 3). Each leaf node in the modified tree T' is

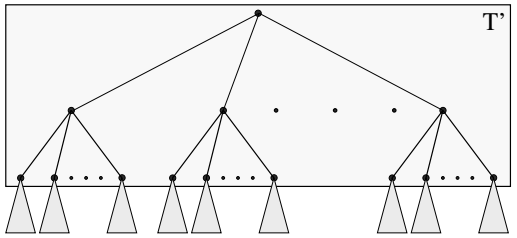

Fig. 3. Partitioning the oct-tree among processors: subtrees at a selected level represent subdomains that are allocated to different processors.

assigned to a different processor. The leaf nodes of T' are the roots of the oct-trees of their respective subdomains. A processor is required to compute the multipole and local coefficients for the oct-tree assigned to it. The computations involving coefficients of parent and child nodes within a processor's oct-tree do not require communication between processors. To compute local coefficients from multipole coefficents, a node requires data from its neighboring nodes which may be resident on different processors. The resulting communication is proportional to the boundary of a subdomain, and may be much less than the useful computation. The coefficients for internal nodes of T' can be computed by a single processor without significant loss in parallel performance of the overall algorithm. To improve the parallel efficiency further, the coefficients for T' can be computed by a smaller set of processors. While load balance can be improved by assigning multiple subdomains to each processor, care must be taken to assign contiguous subdomains to a processor to reduce communication requirements. Other schemes for parallelizing FMM have been presented in [6,10].

5. Experiments

The preconditioned iterative solver outlined earlier can be implemented efficiently on a shared-memory multiprocessor machine. This section presents the results of experiments to study the parallel performance of a multiprocessor implementation of the algorithm. The experiments are organized into three sets. The first set of experiments presents the parallel efficiency of the code on three benchmark problems. The second set of experiments provides an insight into the scalability of the implementation. The third set demonstrates the effect of multipole degree on the parallel performance of the code. These experiments were conducted on a 128-processor SGI Origin2000 with 250 MHz clock speed at the National Center for Supercomputer Applications (NCSA) at the University of Illinois. OpenMP directives were used to parallelize the code and the operating system was allowed to schedule and manage the resulting threads. We report the time spent in the solver (in seconds) as well as the speedup which is defined as the speed improvement obtained over a single processor.

5.1. Examples

The benchmark problems presented in this section are intended to show the parallel performance of the code for realistic problems. The ground plane problem models the ground plane in VLSI circuits that is used to provide a uniform ground to all the interconnects in any of the chips. The spiral inductor problem is a complicated example consisting of a coil shaped conductor which is used in electromagnetic circuitry such as that seen in magnetic access cards. The three-dimensional problem with overlapping two-dimensional panels represents a cross-over of interconnects in typical VLSI circuits.

5.1.1. Ground plane

The first benchmark problem requires computing the inductance of a 1 cm $\times$ 1 cm ground plane. A unit inflow current from the left bottom corner and a unit outflow current from right bottom corner is considered (see Fig. 1). A uniform two-dimensional mesh of size 256×256 is used to discretize the plane. Each filament has length 2^{-8} cm. The width of each filament is one-third of its length, and thickness is 2^{-13} cm. A tolerance of 10^{-3} was specified on the relative residual norm of the preconditioned GMRES method.

Table 2 shows that the speedup increases as the multipole degree is increased. The increase in speedup with multipole degree is due to the increase in computation without corresponding growth in the communication. Even for a relatively small problem, the code achieves 83% efficiency on 64 processors.

5.1.2. Spiral inductor

The spiral conductor problem consists of a coil enclosed in a 1 cm $\times$ 1 cm square region. The conductor is discretized with a uniform two-dimensional mesh similar to the ground plane (see Fig. 4).

Table 3 shows the speedup obtained for a spiral inductor with filament length of 2^{-8} cm. Filament width and thickness are identical to the previous experiment. In spite of the non-uniform point distribution of the problem, the code achieves high

Table 2
Parallel performance on the ground plane problem

No. of processors	Multipole degree					
	1		2		4	
	Time (s)	Speedup	Time (s)	Speedup	Time (s)	Speedup
2	171.57	1.94	641.87	1.93	4476.46	1.99
4	103.33	3.23	336.49	3.68	2251.45	3.96
8	54.34	6.14	174.85	7.08	1167.5	7.63
16	36.36	9.17	91.27	13.55	604.19	14.74
32	20.73	16.09	51.34	24.09	319.85	27.84
64	16.54	20.16	30.35	40.76	166.82	53.39

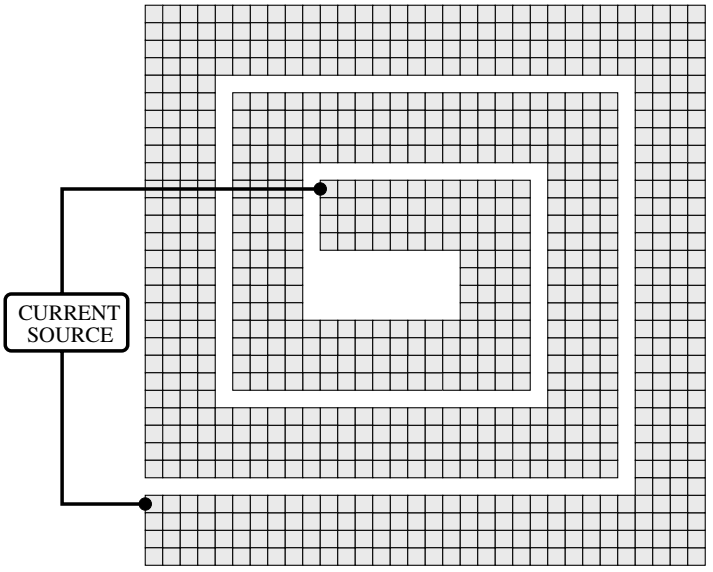

Fig. 4. Spiral inductor.

parallel efficiency. The superlinear performance for higher multipole degree is attributed to the cache-friendly computation for multipole and local coefficients.

5.1.3. Overlapping panels

We consider the problem of determining the impedance of four panels in a three-dimensional configuration shown in Fig. 5. The panels are 1 cm long and 0.25 cm wide. The horizontal separation of panels in a plane is 0.25 cm. The vertical distance (δ) between the two panels is 0.25 cm. Each panel is discretized by a two-dimensional mesh as shown in Fig. 5. The filament width and thickness are identical to the previous experiments.

Table 4 shows the speedup for overlapping panels problem with each panel discretized into a 256×64 mesh of filament length of 2^{-8} cm. The speedup obtained

Table 3
Parallel performance on the spiral conductor problem

No. of processors	Multipole degree					
	1		2		4	
	Time (s)	Speedup	Time (s)	Speedup	Time (s)	Speedup
2	223.72	1.84	791.24	1.93	6715.93	1.99
4	123.91	3.33	426.10	3.58	3365.33	3.97
8	63.07	6.53	211.38	7.21	1731.77	7.72
16	40.27	10.23	115.08	13.25	905.56	14.75
32	23.82	17.29	62.24	24.50	475.27	28.11
64	18.98	21.71	37.20	40.98	208.30	64.14

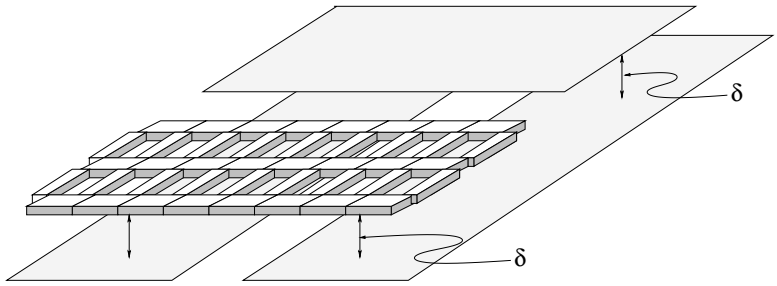

Fig. 5. Four panels in an overlapping 3D configuration.

Table 4
Parallel performance on the overlapping panels problem

No. of processors	Multipole degree					
	1		2		4	
	Time (s)	Speedup	Time (s)	Speedup	Time (s)	Speedup
2	220.02	1.78	840.00	1.55	5716.84	1.99
4	117.02	3.34	338.17	3.84	2870.12	3.95
8	62.45	6.26	183.28	7.09	1465.16	7.75
16	41.71	9.37	99.52	13.05	757.35	14.98
32	21.95	17.80	55.61	23.36	394.47	28.77
64	20.03	19.51	33.16	39.18	174.99	64.85

by the code resembles the previous examples, showing that parallel performance of the code does not diminish for three-dimensional problems.

5.2. Scalability

The performance of the code on the ground plane problem and the overlapping panels problem is used to study the scalability of the algorithm. Larger problem in-

Table 5
Parallel performance for the ground plane problem on different mesh sizes

No. of processors	Mesh size					
	128×128		256×256		512×512	
	Time (s)	Speedup	Time (s)	Speedup	Time (s)	Speedup
2	928.67	1.98	4476.46	1.99	25132.0	1.99
4	465.88	3.95	2251.45	3.96	10376.8	4.81
8	247.76	7.43	1167.5	7.63	5284.24	9.45
16	132.95	13.85	604.19	14.74	2717.67	18.37
32	76.79	23.97	319.85	27.84	1392.07	35.85
64	44.24	41.62	166.82	53.39	713.45	69.96

stances are obtained by refining the mesh. These experiments used degree 4 multipoles. Tables 5 and 6 show speedup for the ground plane and overlapping panels, respectively, for meshes as large as 512×512. For a fixed problem size, the speedup grows almost linearly with the number of processors for larger problems. The slower growth in speedup on smaller problems is partly due to the increase in size of T' which determines the serial component in this implementation. When increasing processors, efficiency can be recovered by allowing the problem to increase as well. This

Table 6
Parallel performance for the overlapping panels problem on different mesh sizes

No. of processors	Panel mesh size					
	128×32		256×64		512×128	
	Time (s)	Speedup	Time (s)	Speedup	Time (s)	Speedup
2	1127.7	1.98	5716.84	1.99	26687.2	1.99
4	565.58	3.96	2870.12	3.95	11286.5	4.70
8	292.64	7.64	1465.16	7.75	5525.52	9.59
6	151.31	14.78	757.35	14.98	2800.73	18.93
32	90.92	24.60	394.47	28.77	1440.41	36.81
64	54.48	41.06	174.99	64.85	734.11	72.21

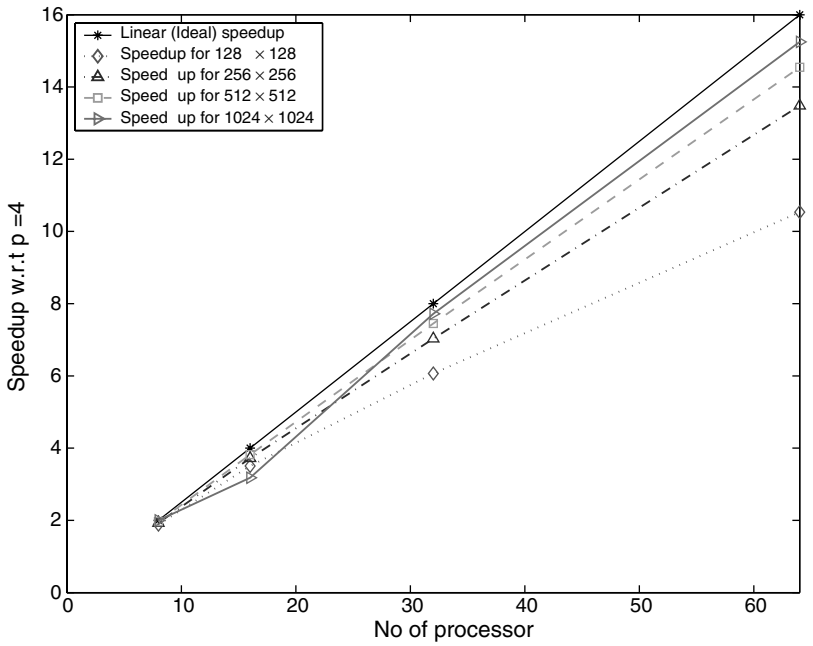

Fig. 6. Speedup w.r.t four processors for the ground plane problem on different mesh sizes.

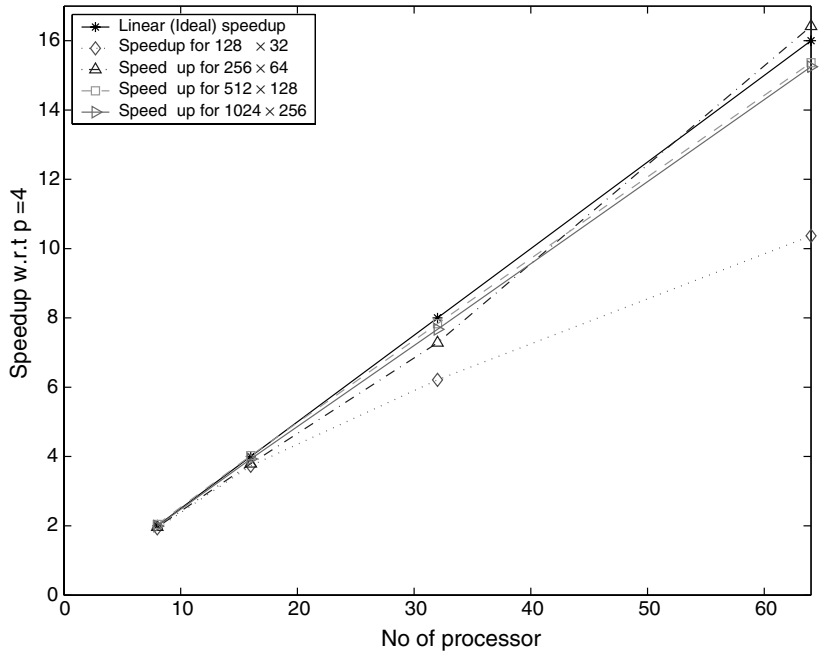

Fig. 7. Speedup w.r.t. four processors for the overlapping panels problem on different mesh sizes.

is evident from Figs. 6 and 7 that show the parallel performance of the code on problem sizes as large as 1024×1024. Since it was not feasible to run the largest instance on fewer than 4 processors, these figures show the speed improvement in time over 4 processors for all problem instances.

5.3. Effect of multipole degree

When using d degree multipoles, the number of multipole and local coefficients at each node is d^2 and the cost of computing these coefficients is proportional to d^4.

Table 7
Parallel performance of the ground plane problem for different choices of multipole degree

Multipole degree	No. of processors						Error (%)
	16		32		64		
	Time (s)	Speedup	Time (s)	Speedup	Time (s)	Speedup	
1	36.36	9.17	20.73	16.09	16.54	20.16	2
2	91.27	13.55	51.34	24.09	30.35	40.76	9×10^{-2}
4	604.19	14.74	319.85	27.84	166.82	53.39	7×10^{-3}
6	2298.26	18.03	1194.71	34.69	642.39	64.51	9×10^{-4}

Table 8
Parallel performance of the overlapping panels problem for different choices of multipole degree

Multipole degree	No. of processors						Error %
	16		32		64		
	Time (s)	Speedup	Time (s)	Speedup	Time (s)	Speedup	
1	41.71	9.37	21.95	17.80	20.03	19.51	1
2	99.52	13.05	55.36	23.26	33.16	39.18	1×10^{-2}
4	757.35	14.98	394.47	28.77	174.99	64.85	8×10^{-3}
6	2325.14	18.67	1212.64	35.80	642.42	67.58	4×10^{-4}

Increase in multipole degree has dual benefits. In addition to the increase in accuracy of the approximation, the parallel performance improves significantly due to a rapid growth in the computation. The speedup often exhibits superlinear behavior due to the cache-friendly nature of these computations. Tables 7 and 8 show the effect of increasing multipole degree on the ground plane and overlapping panels problems, respectively. The column marked error shows the error w.r.t. the solution obtained with degree 8 multipoles which is considered to be accurate for these experiments.

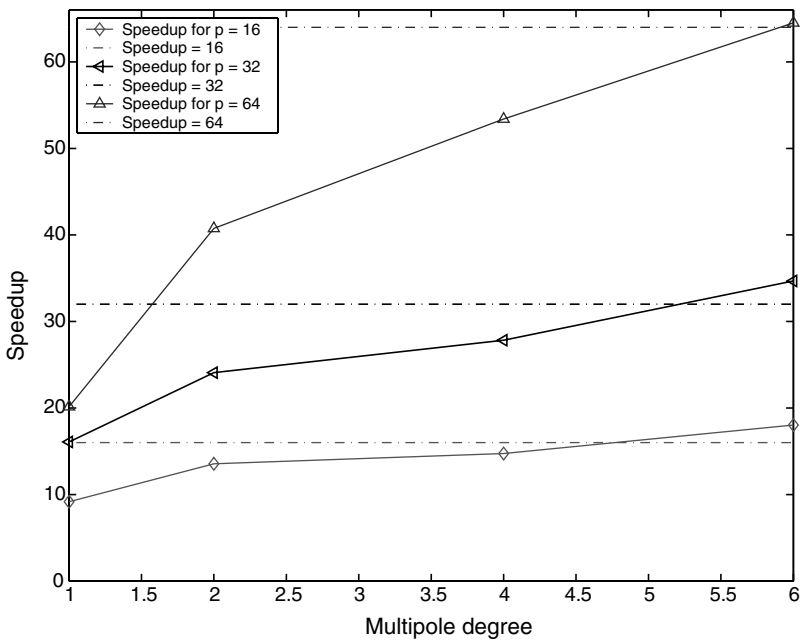

Fig. 8. Effect of multipole degree for ground plane conductor. The horizontal lines indicate the maximum theoretical speedup.

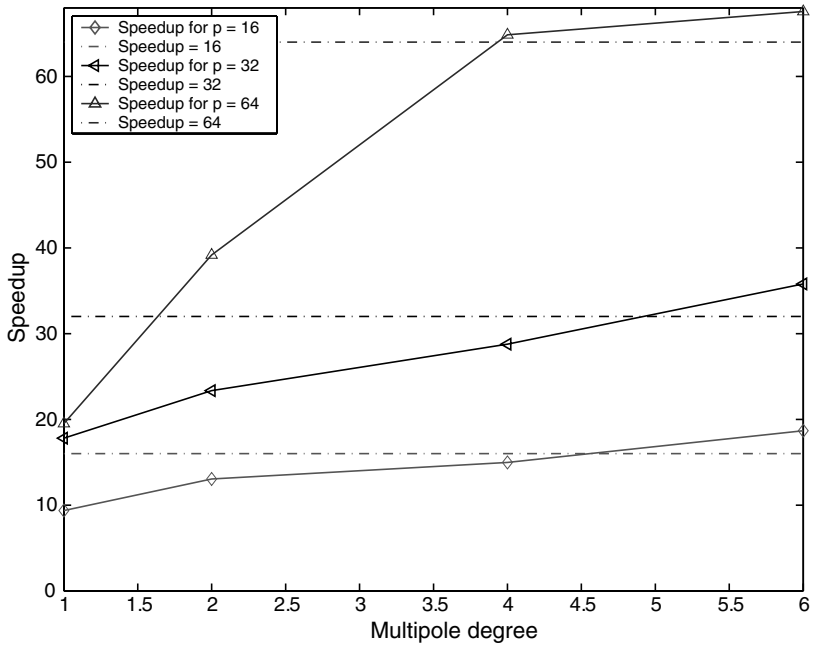

Fig. 9. Effect of multipole degree for overlapping panels. The horizontal lines indicate the maximum theoretical speedup.

Figs. 8 and 9 show that for a fixed problem size, the speedup grows with multipole degree to attain the maximum speedup.

6. Conclusions

In this paper, we presented a preconditioned iterative method for solving the dense linear systems that arise in the inductance extraction problem in VLSI circuit design. The approach uses discrete solenoidal basis functions to obtain an equivalent reduced system which is solved by the preconditioned GMRES method. Matrix–vector products with the dense coefficient matrices as well as preconditioners are computed via hierarchical approximations. We outlined parallelization schemes for the iterative solver with particular emphasis on the hierarchical approximations. Benchmark experiments were presented to show that the implementation achieves high parallel efficiency on the SGI Origin2000 shared-memory multiprocessor. These experiments exhibit almost linear speedups on up to 64 processors for several instances of two- and three-dimensional problems with over a one million unknowns. The results also indicate that higher parallel efficiency can be achieved by increasing the multipole degree d of the FMM. Since the computational cost is proportional to d^4, a larger value of d is recommended only when higher accuracy is desired.

Acknowledgements

This work has been supported in part by NSF under the grants NSF-CCR 9984400 and NSF-CCR 0113668, and by the Texas Advanced Technology Program grant 000512-0266-2001.

References

[1] A. Appel, An efficient program for many-body simulation, SIAM Journal on Scientific and Statistical Computing 6 (1985) 85–103.
[2] J. Barnes, P. Hut, A hierarchical O($n \log n$) force calculation algorithm, Nature 324 (1986) 446–449.
[3] L. Greengard, The Rapid Evaluation of Potential Fields in Particle Systems, The MIT Press, Cambridge, MA, 1988.
[4] F. Grover, Inductance Calculations, Working Formulas and Tables, Dover, New York, 1962.
[5] M. Kamon, M.J. Tsuk, J. White, FASTHENRY: A multipole-accelerated 3D inductance extraction program, IEEE Transactions on Microwave Theory and Techniques 42 (1994) 1750–1758.
[6] J. Leathrum Jr., J. Board Jr., The parallel fast multipole algorithm in three dimensions, Technical Report, Department of Electrical Engineering, Duke University, April 1992.
[7] H. Mahawar, V. Sarin, W. Shi, A solenoidal basis method for efficient inductance extraction, in: Proceedings of the IEEE Design Automation Conference, New Orleans, LA, 2002.
[8] V. Rokhlin, Rapid solution of integral equations of classical potential theory, Journal of Computational Physics 60 (1985) 187–207.
[9] Y. Saad, M.H. Schultz, GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems, SIAM Journal on Scientific and Statistical Computing 7 (1986) 856–869.
[10] J.P. Singh, C. Holt, T. Totsuka, A. Gupta, J.L. Hennessy, Load balancing and data locality in hierarchical N-body methods, Journal of Parallel and Distributed Computing 27 (1995) 118–141.