PARALLEL ALGORITHMS FOR INDUCTANCE EXTRACTION

A Dissertation

by

HEMANT MAHAWAR

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

May 2006

Major Subject: Computer Science

PARALLEL ALGORITHMS FOR INDUCTANCE EXTRACTION

A Dissertation

by

HEMANT MAHAWAR

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Approved by:

Chair of Committee,	Vivek Sarin
Committee Members,	Weiping Shi
	Valerie Taylor
	Nancy Amato
Head of Department,	Valerie Taylor

May 2006

Major Subject: Computer Science

ABSTRACT

Parallel Algorithms for Inductance Extraction. (May 2006)

Hemant Mahawar, B.Tech., Indian Institute of Technology;

M.C.S., Texas A&M University

Chair of Advisory Committee: Dr. Vivek Sarin

In VLSI circuits, signal delays play an important role in design, timing verification and signal integrity checks. These delays are attributed to the presence of parasitic resistance, capacitance and inductance. With increasing clock speed and reducing feature sizes, these delays will be dominated by parasitic inductance. In the next generation VLSI circuits, with more than millions of components and interconnect segments, fast and accurate inductance estimation becomes a crucial step.

A generalized approach for inductance extraction requires the solution of a large, dense, complex linear system that models mutual inductive effects among circuit elements. Iterative methods are used to solve the system without explicit computation of the system matrix itself. Fast hierarchical techniques are used to compute approximate matrix-vector products with the dense system matrix in a matrix-free way. Due to unavailability of system matrix, constructing a preconditioner to accelerate the convergence of the iterative method becomes a challenging task.

This work presents a class of parallel algorithms for fast and accurate inductance extraction of VLSI circuits. We use the solenoidal basis approach that converts the linear system into a reduced system. The reduced system of equations is solved by a preconditioned iterative solver that uses fast hierarchical methods to compute products with the dense coefficient matrix. A Green's function based preconditioner is proposed that achieves near-optimal convergence rates in several cases. By formulating the preconditioner as a dense matrix similar to the coefficient matrix, we are able to use fast hierarchical methods

for the preconditioning step as well. Experiments on a number of benchmark problems highlight the efficient preconditioning scheme and its advantages over FastHenry.

To further reduce the solution time of the software, we have developed a parallel implementation. The parallel software package is capable of analyzing interconnects configurations involving several conductors within reasonable time. A two-tier parallelization scheme enables mixed mode parallelization, which uses both OpenMP and MPI directives. The parallel performance of the software is demonstrated through experiments on the IBM p690 and AMD Linux clusters. These experiments highlight the portability and efficiency of the software on multiprocessors with shared, distributed, and distributed-shared memory architectures.

To my mother, to whom I owe everything

ACKNOWLEDGMENTS

When I came to Texas A&M University in Fall 2000, I was very apprehensive about my Master's degree. I never dreamt that I will pursue a Ph.D. one day. I was not even sure what was in store for me in a new place, both academically and otherwise. At the end of all these years, I can look back and say that I thoroughly enjoyed my time here. My stay has been wonderful, largely due to a set of people who encouraged me, shared my joys and sorrow, and provided me with an overall conducive ambiance.

First and foremost, my sincere thanks to my advisor, Dr. Vivek Sarin, for his invaluable guidance, support and encouragement. His critical evaluations have been instrumental in shaping up my research skills and independent thinking. The amount of freedom that I got while working with him has definitely left me with a satisfying and memorable experience. I have learned a great many things from him and consider myself fortunate to have been one of his first students. My other exploits, namely system administration, in the research lab has been largely possible due to Dr. Sarin's confidence. The freedom given to mess with the workstations and his initiatives to teach me the basics have come a long way.

Working with Dr. Weiping Shi has been a great experience. His ability to present the most complex concepts in a simple manner has helped me understand a lot of new concepts in a short duration. His words of encouragement has often revitalized me in moments of despair. I would also like to thank Dr. Nancy Amato and Dr. Valerie Taylor for agreeing to serve on my committee and for their helpful comments and constructive criticism.

I am thankful to my research lab mates Juttu, Kasthuri, Meiqui, Radhika, Bheem, Thomas and Xue for bearing with me all these years. I am also extremely grateful to all my friends who were always around. They have provided me with an entertaining and relaxing time to make my graduate life memorable. With people like Carlos, Anup, Kasthuri, Radhika, Bheem, Thomas, Vaddi and Vivek to name a few, I missed my family the least.

Numerous discussions with Carlos, Kasthuri and Thomas, ranging from technical interactions to cricket match analysis have always served as a great way to break the monotone of the day. I thank the Computer Science Department staff for taking care of all administrative issues. I would especially miss the love and affection of Cher, Elena and Kathy.

I am deeply indebted to my family members for their love and moral support throughout my life. Nothing would have been possible without the love, support and dedication of my mother. I am what I am because of her efforts. Lastly, one person needs a special mention - Radhika. She has worked with me as a colleague, provided me with an unfading friendship and is sharing my joys and sorrows as life partner. Her support and encouragement throughout these years have made my stay in College Station a rewarding experience.

TABLE OF CONTENTS

Page

CHAPTER Page

LIST OF TABLES

TABLE Page

LIST OF FIGURES

FIGURE Page

CHAPTER I

INTRODUCTION

A. Motivation

In VLSI circuits, signal delays play an important role in design, timing verification and signal integrity checks. These delays are attributed to the presence of parasitic resistance (R), capacitance (C) and inductance (L). Among these parasitic components, primarily capacitance and inductance are functions of operational frequency. As a result of newer technology that uses thicker copper wires, the influence of parasitic resistance has decreased. On the other hand, with operational frequency of modern VLSI circuits approaching gigahertz (GHz) range and shrinking feature sizes, parasitic inductance will have dominating effect on signal delays. For the next generation VLSI circuits with more than millions of components and interconnect segments, there is a significant need for fast and accurate inductance extraction software.

In a VLSI circuit, the changes in current flow create a varying magnetic field that leads to inductive coupling among the different components. This effect is more pronounced when these components are in close physical proximity. Chips designed with sub-micron VLSI technology are prone to parasitic inductive effects because of the tightly packed components. The property of an electrical circuit to resist change in its own current due to the presence of associated magnetic field is called *inductance*. Self inductance of a conductor refers to the impedance offered to current flows by the induced magnetic field due to its own current. Mutual inductance between a pair of conductors refers to the impedance in one conductor due to current flow in the other. The process of estimating these inductive couplings among different components of VLSI circuits is called *inductance extraction*.

The journal model is SIAM Journal of Scientific Computing.

B. Previous Work

There are three types of inductance extraction algorithms: *loop inductance, partial inductance* and *shape based inductance*. The loop inductance algorithms are the most accurate but slowest, while the shape-based algorithms are the least accurate but the fastest. FastHenry [12] is a commonly available software package that computes the loop inductance. Due to its high accuracy, FastHenry is often used as a reference for all other extraction algorithms. Partial inductance was first proposed by Rosa and introduced to circuit design by Ruehli [21]. A number of algorithms have been proposed, such as Krauter [14] and He [10]. Partial inductance algorithms are faster than loop inductance algorithms. However it is shown that partial inductance without current return paths is inaccurate [5]. Shape-based algorithms, such as [13, 25], are fast but inaccurate for complex structures. In this work, we study the extraction of loop inductance of 3D electrical conductors.

A quasi-static approach is often used to compute the parasitic inductance for a set of conductors at a particular frequency. To estimate inductance among a set of conductors in a particular configuration, one needs to determine the current in each conductor under appropriate equilibrium conditions. The general technique is to discretize the surface of each conductor by a uniform two-dimensional mesh that represents a network of smaller conductors or filaments, and a linear system of equations is solved to determine the inductive coupling [12]. The linear system is derived from Kirchoff's current and voltage laws that determine the current flow into mesh nodes and the potential drop across filaments, respectively. The potential drop across the end points of a filament is due to its own resistance and due to the inductive coupling with other filaments. The resulting linear system consists of both sparse and dense sub-matrices. Kirchoff's current law results in a sparse sub-matrix, while the inductive coupling constitutes the entries of the dense and complex sub-matrix.

The cost of computing and storing the dense submatrix becomes prohibitive as the problem size grows beyond a few thousand filaments. Hence, these systems are often solved by iterative methods such as Generalized Minimal Residual (GMRES) method [22]. At each iteration, a matrix-vector product with the coefficient matrix is required. To avoid the memory and computational penalties of exact matrix-vector product, fast hierarchical schemes, such as the Barnes-Hut method [3] and Fast Multipole Method (FMM) [7, 8], are often used to compute approximate matrix-vector products with the system matrix. These approaches have a trade-off between accuracy and speed. These methods can lead to a *matrix-free* algorithm that does not require explicit computation of the coefficient matrix. The success of the underlying iterative methods depends on the rate of convergence that can be accelerated by preconditioning the system. The preconditioning step transforms the original system into an easier one for the iterative methods. If one uses *matrix-free* algorithms for computing matrix vector product with system matrix, then the task of developing preconditioners becomes complicated due to the unavailability of the coefficient matrix.

FastHenry [12] is a commonly used software package for inductance extraction. It uses the above described approach to compute accurate estimates of a circuit's parasitic inductance. Matrix-vector products are computed efficiently by using FMM. Preconditioners are obtained by approximating the dense coefficient matrix with a sparse matrix that is derived from the FMM hierarchical structure. Although the software is used as a benchmark for accuracy comparison, it has found limited use in the VLSI-CAD community due to the long simulation time and large memory requirements. Since FastHenry is available only for uniprocessor workstations, the size of problems that can be solved is severely limited. Hence, there is significant interest in developing fast and accurate parallel algorithms for inductance extraction of large VLSI circuits.

C. Outline

This work presents a class of parallel algorithms for fast and accurate inductance extraction of VLSI circuits. We proposed a solenoidal basis approach for [20] that represents the filament currents in terms of circular cell currents. This approach converts the linear system into a reduced system with fewer unknowns. The reduced system of equations is solved by a preconditioned iterative solver that uses FMM to compute products with the dense coefficient matrix and the preconditioner. We rely on the characteristics of the system matrix to devise the preconditioner. To handle large problem instances and to further reduce the solution time of the software, we have developed a parallel implementation [15, 16]. A two-tier parallelization scheme enables mixed mode parallelization, which uses both OpenMP and MPI directives. Mixed mode parallelization enables the software to run on shared, distributed and distributed-shared memory machines. Experimental results presented in [15, 20] highlight the preconditioning scheme's effectiveness and the parallel performance of the software. To the best of our knowledge, this is the first parallel software for inductance extraction that can run on various multiprocess machines.

The dissertation is organized as follows: Chapter II describes the discretization of the integral equation formulation for the inductance extraction problem and the associated linear system of equations. Chapter III outlines the solenoidal basis method that represents the filament currents in terms of circular cell currents. This approach transforms the linear system into a reduced system. This is followed by a description of the preconditioning approach in Chapter IV. The preconditioning scheme for the reduced system is very effective in reducing the solution time and memory requirements. Experimental results show the effectiveness of the devised preconditioner. The proposed algorithm is compared with FastHenry in Chapter V. Experiments conducted on a set of benchmark problems demonstrate the superiority of our approach. Chapter VI describes the parallel formulation of the

algorithm and the software implementation details. This chapter also includes an overview of the hierarchical multipole-based methods for computing dense matrix-vector products. We present a set of experiments that show the parallel performance of the software on various multiprocessor systems - from supercomputers to workstation clusters. Concluding remarks are presented in Chapter VII.

CHAPTER II

MATHEMATICAL BACKGROUND

A. Problem Statement

The inductance extraction problem for a set of n_s conductors consist of determining an $n_s \times n_s$ complex impedance matrix $\mathbf{Z}(\omega)$ that denotes pairwise mutual impedance among the conductors at a given frequency ω. The kth column of $\mathbf{Z}(\omega)$ is computed by applying unit current to the kth conductor and zero current to all the remaining conductors. Under this boundary condition, the potential drop across the lth conductor gives the Z_{kl} entry. Solutions to n_s such instances with different boundary conditions yields the complete $\mathbf{Z}(\omega)$.

B. Integral Equation

A number of techniques based on the integral form of Maxwell's equations have been used to model VLSI circuits [4, 21]. Maxwell's equations at steady sinusoidal state are given by:

$$\nabla \times \mathbf{E} = -j\omega\mu\mathbf{H} \tag{2.1}$$

$$\nabla \times \mathbf{H} = j\omega\epsilon\mathbf{E} + \mathbf{J} \tag{2.2}$$

$$\nabla \cdot (\epsilon\mathbf{E}) = \rho \tag{2.3}$$

$$\nabla \cdot (\mu\mathbf{H}) = \mathbf{0} \tag{2.4}$$

where $\mathbf{E}$ is electric field, $\mathbf{H}$ is magnetic field, $\mathbf{J}$ is current density, ω is frequency of operation, μ is the magnetic permeability, ϵ is the electric permittivity and $j = \sqrt{-1}$. Equations (2.1-2.4) describe, respectively, how changing magnetic field produce electric fields (Faraday's law), how currents produce magnetic fields (Ampere's law), how electric

charges produce electric fields (Gauss's law), and the absence of magnetic field. Additionally, by Ohm's law, the electric field within the conductor is related to the current density by:

$$\mathbf{E} = \rho \mathbf{J}. \tag{2.5}$$

where ρ is the resistivity of the material. Applying quasi-static assumption that the displacement current $j\omega\epsilon\mathbf{E}$ is negligible, the divergence of (2.2) yields current conservation:

$$\nabla \cdot \mathbf{J} = \mathbf{0}. \tag{2.6}$$

We wish to eliminate the fields $\mathbf{E}$ and $\mathbf{H}$, and represent the Maxwell's equations in terms of the current density $\mathbf{J}$ and applied voltage only. From (2.4), the magnetic flux can be represented as:

$$\mu\mathbf{H} = (\nabla \times \mathbf{A}) \tag{2.7}$$

where $\mathbf{A}$ is the magnetic vector potential. Using it in conjunction with (2.1), we get:

$$\nabla \times (\mathbf{E} + j\omega\mathbf{A}) = 0.$$

This implies that there exists a scalar potential function $\mathbf{\Phi}$ such that:

$$-\nabla\mathbf{\Phi} = \mathbf{E} + j\omega\mathbf{A}. \tag{2.8}$$

To relate the vector potential $\mathbf{A}$ to the current density $\mathbf{J}$ we use (2.7) and the Coulomb gauge relation: $\nabla \cdot \mathbf{A} = 0$. Under quasi-static assumptions, it converts (2.2) into:

$$-\nabla^2\mathbf{A} = \mu\mathbf{J}.$$

Hence magnetic vector potential $\mathbf{A}$ is represented as:

$$\mathbf{A}(\mathbf{r}) = \frac{\mu}{4\pi} \int_V \frac{\mathbf{J}(\mathbf{r}')}{\|\mathbf{r} - \mathbf{r}'\|} dV' \tag{2.9}$$

where $\mathbf{r}$ and $\mathbf{r}'$ denote three-dimensional position vectors, V is the volume of the conductor, and dV' is the incremental volume with respect to $\mathbf{r}'$.

Substituting (2.5) and (2.9), into (2.8), we get the following integral equation that relates the current density $\mathbf{J}(\mathbf{r})$ and potential $\mathbf{\Phi}(\mathbf{r})$ at steady state:

$$\rho \mathbf{J}(\mathbf{r}) + \frac{j\omega\mu}{4\pi} \int_V \frac{\mathbf{J}(\mathbf{r}')}{\|\mathbf{r} - \mathbf{r}'\|} dV' = -\nabla \mathbf{\Phi}(\mathbf{r}). \tag{2.10}$$

Using (2.6) and (2.10), the current density $\mathbf{J}$ and scalar potential $\mathbf{\Phi}$can be computed.

C. Linear System

To obtain a numerical solution of (2.6) and (2.10), each conductor surface is discretized using a uniform two-dimensional mesh (see, e.g., Fig. 1). Current carrying filaments form the edges of the mesh. Given the quasi-static assumption, there is no charge accumulation on the conductor surface. Hence, current density is assumed to be constant within each filament and the current is assumed to flow only along the length of the filament. The vector of filament currents $\mathbf{I}_f$ is related to the vector of potential drop across filament end points $\mathbf{V}_f$ by the following equation, which is a discrete form of (2.10):

$$[\mathbf{R} + j\omega\mathbf{L}]\, \mathbf{I}_f = \mathbf{V}_f, \tag{2.11}$$

where $\mathbf{R}$ is an $n \times n$ diagonal matrix of filament resistances for a mesh with n filaments $f_1, f_2, \ldots f_n$, and $\mathbf{L}$ is an $n \times n$ dense inductance matrix. The kth diagonal element of $\mathbf{R}$ is assigned the value $\rho \cdot l_k / a_k$, where l_k and a_k are the length and cross-sectional area of kth

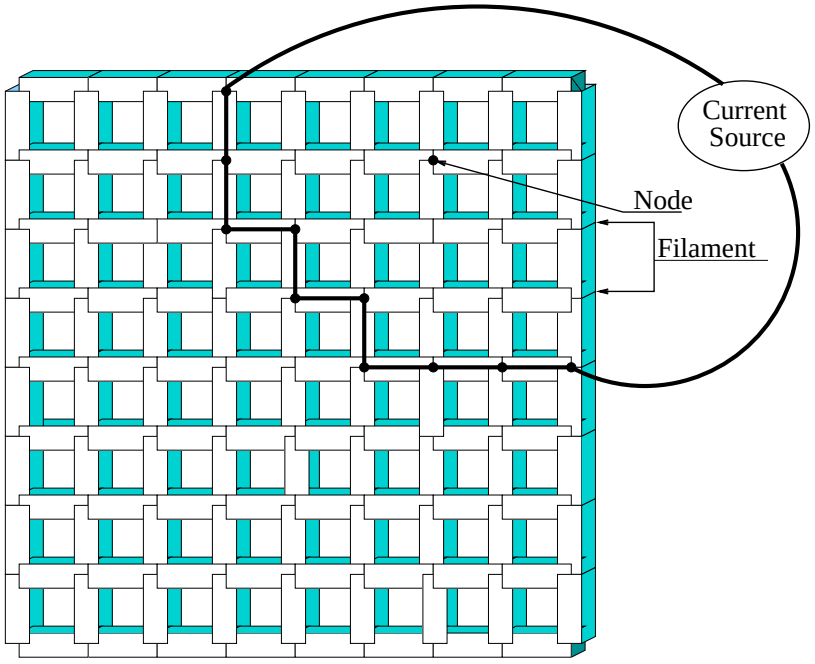

Fig. 1. Discretization of a conductor surface using two-dimensional mesh.

filament, respectively. The entries of the inductance matrix $\mathbf{L}$ are:

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

where $\mathbf{u_k}$ denotes the unit vector along the kth filament, and $\mathbf{r}_k$ and $\mathbf{r}_l$ denote position vectors for points in filaments k and l, respectively. The integral is calculated over the volume of the two filaments.

Kirchoff's current law specifies that the net flow of current is zero at each node of the mesh. It is represented by the following equation:

$$\mathbf{B}^T\mathbf{I}_f = \mathbf{I}_s, \tag{2.12}$$

where $\mathbf{B}^T$is a sparse $m \times n$ branch index matrix of m nodes and n filaments and $\mathbf{I}_s$ is the known branch current vector of length m with non-zero values corresponding to the source currents. The (k, l) entry of branch index matrix has the value -1 if the lth filament

originates at node k, 1 if it terminates at k, and zero otherwise. For a $n_x \times n_y$ array of cells, the entries of $\mathbf{B}^T$ are:

$$\mathbf{B}^T = \left[\begin{array}{ccccc|ccccc} \overbrace{}^{(n_y+1)} & & & & & \overbrace{}^{n_y} & & & & \\ T & & & & & I & & & & \\ & T & & & & -I & I & & & \\ & & T & & & & -I & \ddots & & \\ & & & \ddots & & & & \ddots & I & \\ & & & & T & & & & & -I \end{array}\right],$$

where I is the identity matrix of size $(n_x+1)\times(n_x+1)$ and T is a matrix of size $(n_x+1)\times n_x$ given below:

$$T = \begin{bmatrix} 1 & & & \\ -1 & 1 & & \\ & -1 & \ddots & \\ & & \ddots & 1 \\ & & & -1 \end{bmatrix}.$$

The nodes are numbered from left to right in bottom to top fashion. The filaments are numbered in a similar fashion, first the horizontal filaments and than the vertical ones.

Potential drop across filaments $\mathbf{V}_f$ can be expressed in terms of unknown node potentials $\mathbf{V}_n$ as follows:

$$\mathbf{V}_f = \mathbf{B}\mathbf{V}_n. \tag{2.13}$$

Equations (2.11), (2.12), and (2.13) give rise to a linear system of equations that must be solved to determine the unknown filament currents $\mathbf{I}_f$ and node potentials $\mathbf{V}_n$:

$$\begin{bmatrix} \mathbf{R} + j\omega\mathbf{L} & -\mathbf{B} \\ \mathbf{B}^T & 0 \end{bmatrix} \begin{bmatrix} \mathbf{I}_f \\ \mathbf{V}_n \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{I}_s \end{bmatrix}. \tag{2.14}$$

D. Iterative Solvers

The coefficient matrix of the linear system (2.14) consists of sparse and dense sub-matrices. The first diagonal block is dense and the off-diagonal blocks are sparse. A straightforward approach to solve this linear system involves removal of $\mathbf{I}_f$ by a block-step of Gaussian elimination. The resulting system is defined in terms of the unknowns $\mathbf{V}_n$ only, and can be expressed as:

$$\mathbf{B}^T[\mathbf{R} + j\omega\mathbf{L}]^{-1}\mathbf{B}\mathbf{V}_n = \mathbf{I}_s.$$

Even for a few thousands of unknowns, use of direct methods becomes prohibitively expensive due to memory constraints and the size of the system. For such large linear systems, iterative methods such as GMRES are often used. When solving this system by an iterative method, each iteration involves a matrix-vector product with the system matrix. In practice, the system matrix is never computed explicitly. Instead, the matrix-vector product with a vector x is computed as a sequence of three steps:

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

The second step may require an inner iterative scheme, resulting in expensive outer iterations. Moreover, the structure of the coefficient matrix makes it very difficult to precondition the linear system.

The computational cost of iterative methods depends on matrix-vector calculation with the coefficient matrix. The number of operations needed to compute an accurate matrix-

vector product with an $n \times n$ dense matrix is $O(n^2)$. In (2.14), $\mathbf{R}$ and $\mathbf{B}$ both are sparse matrices, hence the cost of matrix-vector product is dominated by the cost of computing product with $\mathbf{L}$. Even for a small problem with few thousands filaments, it is expensive to compute and store $\mathbf{L}$. On the other hand, if the matrix entries are functions of the form $1/r$, approximations to these products can be computed efficiently through *matrix-free* hierarchical based methods. These methods exploit the decaying nature of $1/r$ kernel and can be used to compute approximations for the matrix-vector products with $\mathbf{L}$. These methods include FMM and variants of the Barnes-Hut method. When the filaments are uniformly distributed, FMM requires $O(n)$ operations while multipole-based variants of the Barnes-Hut method require $O(n \log n)$ operations to compute these matrix-vector products. These approaches have a trade-off between accuracy and speed. Higher degree multipoles can be used to reduce the approximation error. However, the computational cost grows proportional to d^4, where d is the multipole degree.

Use of iterative methods to solve a linear system is meaningful only if the underlying iterative method has a fast rate of convergence. With efficient preconditioning of the coefficient matrix, one can accelerate the convergence of underlying iterative scheme. Preconditioning may be considered as a process of transforming a linear system into one that can be solved more efficiently by the iterative process. The use of *matrix-free* hierarchical methods ensures that the coefficient matrix is never constructed. However, construction of preconditioners in the absence of the coefficient matrix turns out to be a formidable challenge.

CHAPTER III

SOLENOIDAL BASIS METHOD

Solenoidal functions are divergence-free functions that satisfy conservation laws such as the Kirchoff's current law in electrical circuits and the mass conservation law in fluid mechanics. These functions have been applied to a variety of engineering applications such as computational fluid dynamics (CFD) etc. [23]. A solenoidal vector field $\mathbf{G}$ satisfies: $\nabla \cdot \mathbf{G} = \mathbf{0}$. If this condition is satisfied, there exists a vector $\mathbf{D}$, such that: $\mathbf{G} = \nabla \times \mathbf{D}$. Equation (2.6) states that the current is a divergence-free vector function. Hence, one can represent current as curl of a vector function. For inductance extraction problem with uniform mesh discretization, it is easy to construct a solenoidal basis that represent the curl operator in the discrete sense.

The second block of (2.14) represents Kirchoff's current law. Since $\mathbf{B}^T$ enforces the current conservation, the null space of $\mathbf{B}^T$ represents a basis for current that obeys Kirchoff's law. Any full-rank matrix $\mathbf{P}$ that satisfies $\mathbf{B}^T\mathbf{P}= \mathbf{0}$, can be used to compute the current vector via the matrix-vector product as follows:

$$\mathbf{I} = \mathbf{P}x.$$

There are several ways to compute a basis for the null space of a matrix. A purely algebraic approach such as QR factorization of $\mathbf{B}^T$ cannot be used to compute $\mathbf{P}$ due to the prohibitive cost of computation and storage of a large dense matrix. However, to construct a sparse basis, observe that a current flow of fixed magnitude along any closed path in the mesh satisfies the constraints imposed by $\mathbf{B}^T$. Figure 2 shows several instances of discrete solenoidal current flows that can be used to construct the solenoidal basis. Each flow consists of a constant amount of current flowing anticlockwise through the four filaments of a

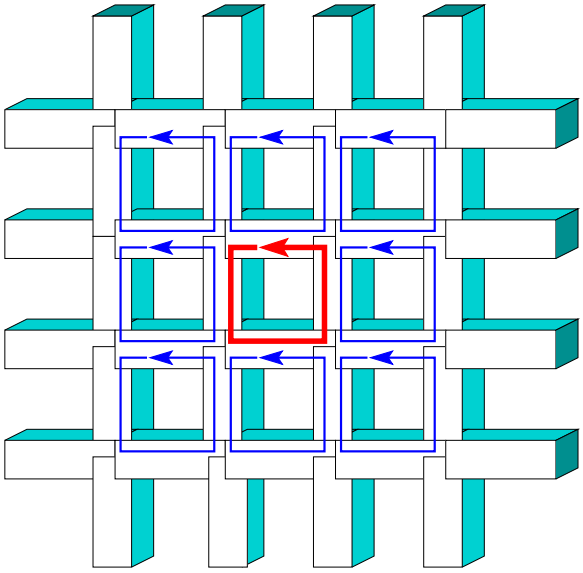

Fig. 2. Examples of solenoidal current flows in a section of a uniform two-dimensional mesh.

cell in the mesh. Since the net inflow of these circular currents into any node is zero, the flows satisfy Kirchoff's current law. The solenoid basis method for (2.14) uses these mesh currents to represent the unknown current $\mathbf{I}_f$. A basis consisting of such functions can be viewed as a *local solenoidal basis*.

A local solenoidal basis for a two-dimensional mesh is a complete basis with linearly independent components. The linear independence of the columns of $\mathbf{P}$ is established by observing that the matrix $\mathbf{P}^T\mathbf{P}$ is the standard two-dimensional Laplace operator matrix. In a uniform two-dimensional mesh of size $n_x \times n_y$, the number of nodes, edges, and cells are given by $m = n_x.n_y$, $n = 2m - n_x - n_y$, and $s = n - m + 1$, respectively. Since the number of cells equals the dimension of the null space of the discrete divergence matrix $\mathbf{B}^T$, it follows that the local solenoidal basis is complete. It may be noted that $\mathbf{B}$ has a rank-deficiency of 1 to allow the potential to vary by a constant. The rank deficiency can be removed by specifying the value of the potential at a single node, and the solenoidal basis can be modified accordingly.

The solenoidal basis matrix $\mathbf{P}$ is an $n \times s$ matrix that is derived from the current flows in the mesh cells. The columns of $\mathbf{P}$ correspond to the cells in the mesh. Each column of $\mathbf{P}$ consists of four non-zero entries that denote the current flow in the cell: 1 indicates a unit current flow along the edge, and -1 indicates a unit current flow opposite to the direction of edge. Construction of the solenoidal basis matrix in this manner ensures that the following condition is satisfied:

$$\mathbf{B}^T\mathbf{P} = 0. \tag{3.1}$$

For an $n_x \times n_y$ array of cells, the entries of $\mathbf{P}^T$ are:

$$\mathbf{P}^T = \left[\begin{array}{cccccc|ccccc} \overbrace{\begin{matrix} I & -I & & & \end{matrix}}^{(n_y+1)} & & & & & & \overbrace{\begin{matrix} W & & & \end{matrix}}^{n_y} & & & \\ & \begin{matrix} I & -I & & \end{matrix} & & & & & & \begin{matrix} W & & \end{matrix} & & \\ & & \ddots & \ddots & & & & & \ddots & & \\ & & & & I & -I & & & & & W \end{array}\right],$$

where I is the identity matrix of size $n_x \times n_x$ and W is a matrix of size $n_x \times (n_x + 1)$ given below:

$$W = \begin{bmatrix} 1 & -1 & & & \\ & 1 & -1 & & \\ & & \ddots & \ddots & \\ & & & 1 & -1 \end{bmatrix}.$$

The meshes are numbered from left to right in bottom to top fashion. The filaments are numbered in a similar fashion, first the horizontal filaments and than the vertical ones.

The linear system (2.14) must be transformed before one can use the solenoidal basis method. The first step is to determine a particular current vector $\mathbf{I}_p$ that satisfies the con-

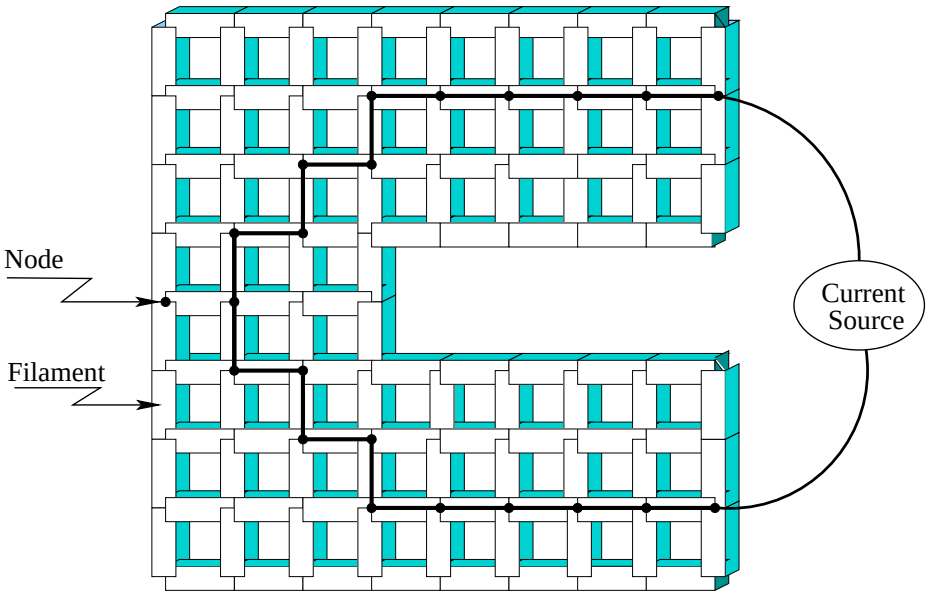

Fig. 3. Current flow along an arbitrary path in the mesh can be used to satisfy the constraints imposed by the external current source. The bold line indicates a path for current that satisfies boundary conditions.

straints imposed by the external current source. The vector $\mathbf{I}_p$ represents the current flow along an arbitrary path between the nodes where the external source is connected. The current vector $\mathbf{I}_p$ can easily be found by a number of techniques. For instance, when the known branch current has unit magnitude, one can assign a unit current to filaments on an arbitrary path from the node with input source current to the node with output source current (see Fig. 3). This approach can be extended to more general boundary conditions in a straightforward manner. By splitting the current $\mathbf{I}_f$ into a particular current $\mathbf{I}_p$ and an unknown current $\mathbf{I}$, the linear system (2.14) can be transformed to an equivalent system with a different right hand side:

$$\begin{bmatrix} \mathbf{R}+j\omega\mathbf{L} & -\mathbf{B} \\ \mathbf{B}^T & 0 \end{bmatrix} \begin{bmatrix} \mathbf{I} \\ \mathbf{V}_n \end{bmatrix} = \begin{bmatrix} \mathbf{F} \\ \mathbf{0} \end{bmatrix}, \tag{3.2}$$

where

$$\mathbf{I} = \mathbf{I}_f - \mathbf{I}_p, \qquad \mathbf{F} = -\left[\mathbf{R} + j\omega\mathbf{L}\right]\mathbf{I}_p.$$

The difference between (2.14) and (3.2) is that the first system satisfies current boundary conditions whereas the second system satisfies voltage boundary conditions.

The next step is to represent the unknown current $\mathbf{I}$ in the solenoidal basis:

$$\mathbf{I} = \mathbf{P}x,$$

where x is an unknown vector of size s. From (3.1), it follows that $\mathbf{I}$ satisfies the divergence-free constraints imposed by the second block of equations in (3.2). By restricting the unknown current to the solenoidal subspace, the linear system (3.2) can be transformed to the following system:

$$\left[\mathbf{R} + j\omega\mathbf{L}\right]\mathbf{P}x - \mathbf{B}\mathbf{V}_n = \mathbf{F}.$$

The vector of unknown node potentials $\mathbf{V}_n$ can be eliminated by multiplying the system with $\mathbf{P}^T$ from the left:

$$\mathbf{P}^T\left[\mathbf{R} + j\omega\mathbf{L}\right]\mathbf{P}x = \mathbf{P}^T\mathbf{F}. \tag{3.3}$$

The above system is a *reduced* linear system of order s that must be solved to determine x. Since the coefficient matrix is never computed explicitly, an iterative method such as GMRES must be used to solve the system. Once x has been computed, the filament current vector $\mathbf{I}_f$ and the vector of unknown potential differences across the filaments $\mathbf{V}_f$ can be computed as follows:

$$\mathbf{I}_f = \mathbf{I}_p + \mathbf{P}x, \qquad \mathbf{V}_f = \left[\mathbf{R} + j\omega\mathbf{L}\right]\mathbf{I}_f.$$

Table I. The sizes of the original and reduced systems arising in the inductance extraction of a ground plane.

Mesh Size	Nodes (m)	Filaments (n)	Unknowns ($n+m$)	Solenoidal Functions (s)
32×32	1089	2112	3201	1024
64×64	4225	8320	12545	4096
128×128	16641	33024	49665	16384
256×256	66049	131584	197633	65536

When a unit current is applied by the external source, the impedance between any pair of nodes is equal to the potential difference between the nodes. This potential difference can be calculated by adding the potential differences across the filaments on an arbitrary path connecting the nodes.

A. Benefits of Solenoidal Basis

The transformation of the linear system (3.2) to the reduced system (3.3) has several advantages. The number of unknowns reduces considerably whenever two-dimensional discretization is employed. Table I shows the number of unknowns in a ground plane problem that involves computing the self impedance of a square conductor. The use of a local solenoidal basis results in a sparse matrix $\mathbf{P}$ that is amenable to efficient matrix-vector product computations. Furthermore local nature of $\mathbf{P}$ ensures that operations such as computation and storage of $\mathbf{P}$ and matrix-vector products with $\mathbf{P}$ can be implemented efficiently in parallel. Matrix-free implementations are also possible since explicit construction of $\mathbf{P}$ is not necessary. The local solenoidal basis has another property that is useful in constructing preconditioners for the reduced system in (3.3). The application of $\mathbf{P}$ and $\mathbf{P}^T$ to vectors is analogous to computing the discrete curl of a function.

CHAPTER IV

PRECONDITIONING SCHEME

Even for the reduced system (3.3), the use of direct methods to compute the unknown cell current becomes prohibitively expensive for modest sized problems. Direct methods suffer from high computational costs and large memory requirements. To overcome these hurdles, iterative methods are used. Use of iterative methods to solve a linear system is meaningful only if the underlying iterative method has a fast rate of convergence.

The rate of convergence of iterative methods is related to the spectral properties of the system matrix. For instance, a large separation between the smallest and largest eigenvalues of a matrix often results in a large number of iterations required for convergence. Preconditioning is a process of transforming a linear system into one that has more favorable spectral properties. The linear system $\mathbf{A}x = b$ may be preconditioned from the right side by a matrix $\mathcal{M}$ as shown below:

$$\mathbf{A}\mathcal{M}y = b, \qquad x = \mathcal{M}y. \tag{4.1}$$

The transformed system is solved by an iterative method. The coefficient matrix $\mathbf{A}\mathcal{M}$ is never computed explicitly. Instead, each iteration now requires an additional preconditioning step that involves computing a matrix-vector product with $\mathcal{M}$.

A preconditioning approach is advantageous only if the overall time to compute the solution is reduced. For this to happen, the preconditioner must be easy to compute, the preconditioning step must be relatively inexpensive, and the matrix $\mathcal{M}$ should be an effective preconditioner that reduces the number of iterations considerably. Preconditioning can be done from the left side by pre-multiplication, from the right side by post-multiplication, or from both sides [22]. A good preconditioner can be characterized in a variety of ways.

In general, clustering of eigenvalues of the preconditioned system can lead to rapid convergence. In many cases, a preconditioned system with a small condition number can be solved in a few iterations. Condition number of a matrix can be estimated by the ratio of the largest singular value to the smallest. A preconditioner is said to be *optimal* when the condition number of the preconditioned system is bounded by a constant. In such a case, iterative methods may converge to the solution in fixed number of iterations regardless of the problem discretization.

It is common to use the symbol $\mathcal{M}^{-1}$ instead of $\mathcal{M}$ in (4.1) to indicate that the preconditioner is an approximation of the matrix $\mathbf{A}$. In such cases, the preconditioning step requires the solution of a linear system with the preconditioner as the coefficient matrix. Such approximations are obtained implicitly by computing incomplete factorizations of $\mathbf{A}$. Although the coefficient matrix in the reduced system (3.3) is not available, one can compute a "sparse" approximation by ignoring interactions between distant filament pairs. An incomplete factorization of this sparse matrix yields L and U factors that can be used to define the preconditioner for the reduced system. FastHenry [12] uses a similar approach in which the matrix is sparsified by several different strategies in order to obtain inexpensive but effective preconditioners. These schemes are describes in more detail in Chapter V. Unfortunately, the sparsification schemes in the package tend to be very slow and have huge memory requirements. This has restricted the use of the software to solving small benchmark problems only.

A. Spectral Analysis of Reduced System

The ability to precondition the reduced system effectively is critical to the success of the solenoidal basis method. The task of designing effective preconditioners is made especially challenging due to the unavailability of $\mathbf{L}$. An effective preconditioning approach can be

Table II. Estimates of the extremal eigenvalues of matrices that form the reduced system for a ground plane problem.

Matrix	$\lambda_{\min}$	$\lambda_{\max}$
$\mathbf{R}$	c	c
$\mathbf{L}$	$O(h)$	$O(1)$
$\mathbf{P}^T\mathbf{R}\mathbf{P}$	$O(h^2)$	$O(1)$
$\mathbf{P}^T\mathbf{L}\mathbf{P}$	$O(h^2)$	$O(h)$
$Re(\mathbf{P}^T\left[\mathbf{R}+j\omega\mathbf{L}\right]\mathbf{P})$	$O(h^2)$	$O(1)$
$Im(\mathbf{P}^T\left[\mathbf{R}+j\omega\mathbf{L}\right]\mathbf{P})$	$O(\omega h^2)$	$O(\omega h)$

developed by analyzing the reduced system (3.3) carefully. The use of local solenoidal flows defined on a uniform two-dimensional mesh for a ground plane (see Fig. 1) provides a basis for this analysis. Consider the matrices that form the reduced system: $\mathbf{R}$ is a diagonal matrix with positive entries, $\mathbf{L}$ is a dense symmetric positive definite (SPD) matrix, and $\mathbf{P}$ is a sparse matrix. The matrices $\mathbf{P}$ and $\mathbf{P}^T$ implement discrete curl operators, and $\mathbf{P}^T\mathbf{P}$ is a two-dimensional discrete Laplace operator. Table II provides estimates of the largest and smallest eigenvalues of these matrices for a discretization with filament length h. To be consistent with physical laws, conductor surfaces must be discretized using filaments with a fixed length-to-width ratio. As a result, the eigenvalues of $\mathbf{R}$ are always constant.

At higher frequencies, the reduced system is dominated by the imaginary part whose condition number is proportional to h^{-1}. The real part dominates at lower frequencies, and the condition number grows proportional to h^{-2}. Figure 4 shows the spectrum of the reduced system (3.3) obtained from the ground plane problem with $\omega = 2\pi\times$10GHz. A uniform two-dimensional mesh is used to discretize the ground plane of size 1cm$\times$1cm. For an $n_x \times n_x$ size mesh, the filament length is $h = 1/n_x$ cm. The eigenvalues of the system lie on a straight line in the complex plane. The condition number of the reduced

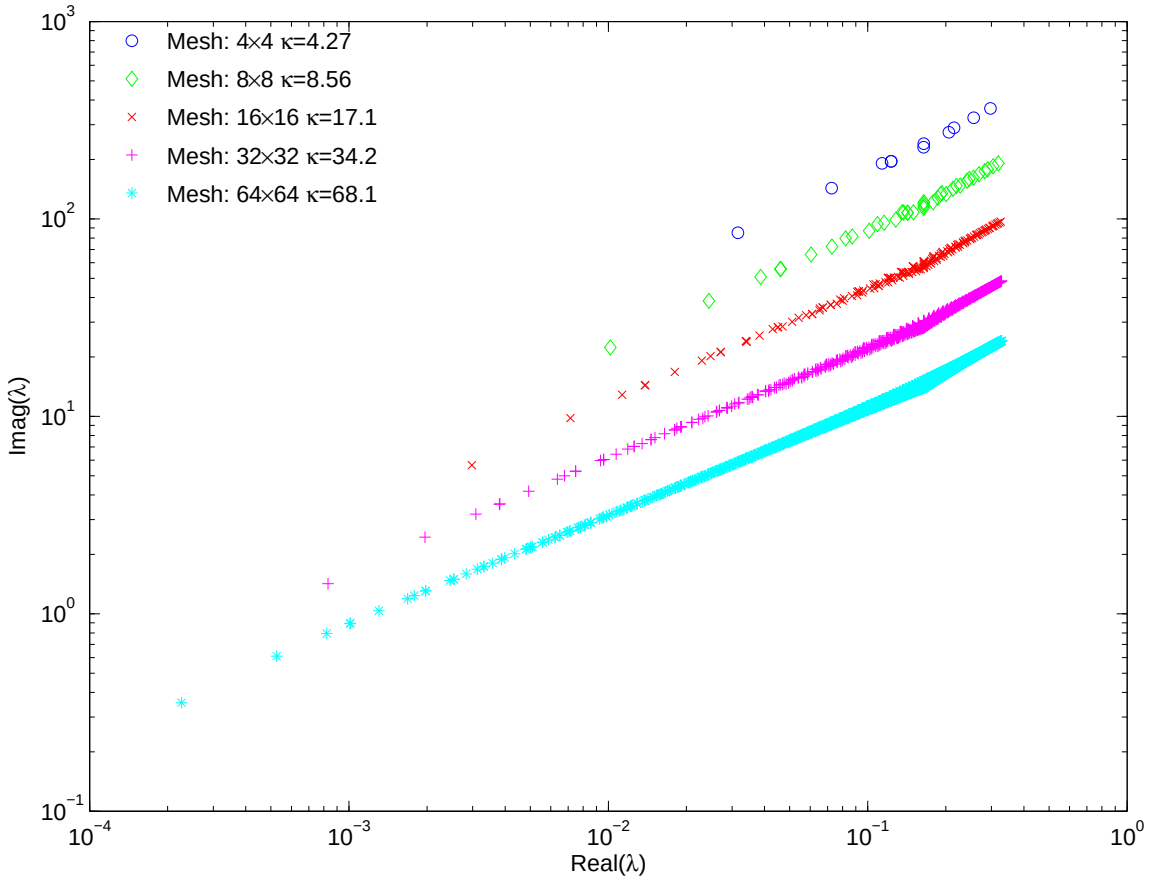

Fig. 4. The spectrum of the reduced system that arises from a uniform discretization of a square ground plane. The condition number of the system is denoted by κ ($\omega = 2\pi \times$ 10GHz).

system doubles every time the mesh is refined.

The entries of the matrix $\mathbf{L}$ are derived from the Green's function for the three-dimensional Laplace operator. While $\mathbf{P}^T\mathbf{P}$ is a two-dimensional Laplace operator with a condition number proportional to h^{-2}, the matrix $\mathbf{P}^T\mathbf{L}\mathbf{P}$ tends to have a condition number that is proportional to h^{-1} only. As shown in Fig. 5, the matrix $\mathbf{L}\mathbf{P}$ is a well-conditioned matrix with a condition number that is nearly independent of h, indicating that $\mathbf{L}$ and $\mathbf{P}$ are approximate "inverse" of each other. To exploit this fact, we express $\mathbf{L}\mathbf{P}$ as shown below:

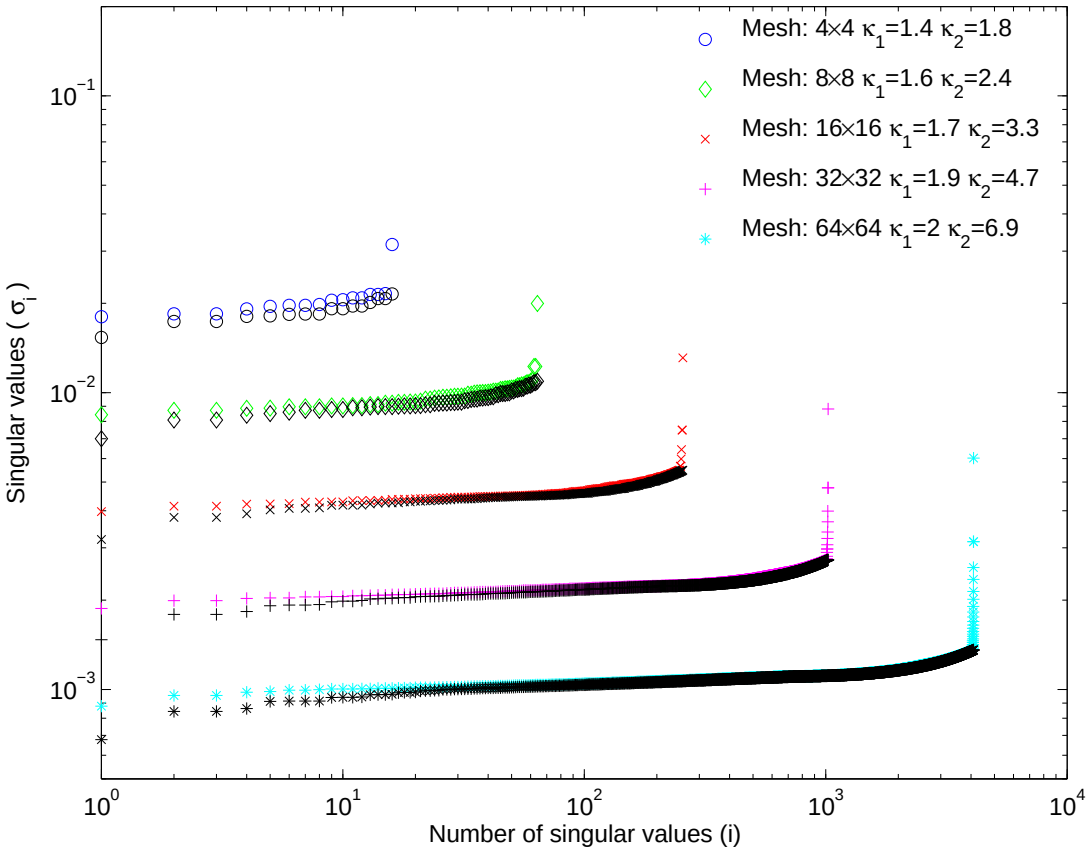

Fig. 5. Singular values of $\mathbf{LP}$ (black) and $\mathbf{P\tilde{L}}$ (color) for the square ground plane problem. The condition numbers of $\mathbf{LP}$ and $\mathbf{P\tilde{L}}$ are denoted by κ_1 and κ_2, respectively ($\omega = 2\pi\times$ 10GHz).

$$\mathbf{LP} = \mathbf{P\tilde{L}} + \hat{\mathbf{P}}\hat{\mathbf{L}},$$

where $\tilde{\mathbf{L}}$ represents the mutual inductance among hypothetical filaments placed at the center of each mesh cell and oriented perpendicular to the mesh plane, $\hat{\mathbf{L}}$ represents the inductive effect of these filaments on the boundary of the conductors, and $\hat{\mathbf{P}}$ updates the boundary filaments. This representation can be viewed as a set of filaments placed at the center of the mesh cells with an additional set of ghost-cells along the boundary. The filaments at the mid-points of the ghost cells carry no current. Since the effect of $\hat{\mathbf{P}}\hat{\mathbf{L}}$ is limited to the

boundary, one can expect $\mathbf{P}\tilde{\mathbf{L}}$ to be a good approximation of $\mathbf{LP}$. Figure 5 shows that a large number of singular values of these two matrices are identical.

The matrix $\tilde{\mathbf{L}}$ is defined as follows:

$$\tilde{\mathbf{L}}_{kl} = \frac{\mu}{4\pi}\frac{1}{a_k a_l}\int_{r_k \in V_k}\int_{r_l \in V_l}\frac{1}{\|\mathbf{r}_k - \mathbf{r}_l\|_2}dV_k dV_l.$$

The matrix element $\tilde{\mathbf{L}}_{kl}$ equals the mutual inductance between a pair of parallel filaments placed at the centers of cells k and l. At high frequencies, when the imaginary part of the reduced system dominates, $\tilde{\mathbf{L}}$ can be used as a preconditioner for the reduced system. Since

$$\mathbf{P}^T\mathbf{L}\mathbf{P}\tilde{\mathbf{L}} \approx \mathbf{P}^T\mathbf{L}^T\mathbf{L}\mathbf{P},$$

the preconditioner is expected to yield a well-conditioned system.

Since $\tilde{\mathbf{L}}$ represents an approximate inverse of $\mathbf{P}$, using the characteristics of the reduced system, we propose the following efficient preconditioning scheme for inductance extraction:

$$\mathcal{M} = \tilde{\mathbf{L}}\left[\tilde{\mathbf{R}} + j\omega\tilde{\mathbf{L}}\right]^{-1}\tilde{\mathbf{L}}, \tag{4.2}$$

where $\tilde{\mathbf{R}}$ is a diagonal matrix of resistance to mesh currents. At each iteration, the preconditioning step consists of the matrix-vector product $z = \mathcal{M}r$ that can be computed in the following three steps:

$$u = \tilde{\mathbf{L}}r, \qquad v = \left[\tilde{\mathbf{R}} + j\omega\tilde{\mathbf{L}}\right]^{-1}u, \qquad z = \tilde{\mathbf{L}}v.$$

The matrix-vector products in the first and third steps use approximate hierarchical techniques identical to those used for $\mathbf{L}$. The second step is implemented via an inner iterative solver that is used to solve the system

$$\left[\tilde{\mathbf{R}} + j\omega\tilde{\mathbf{L}}\right] v = u$$

to obtain v. At low and high frequencies, one can use the following approximations to the preconditioner without any significant change in the rate of convergence:

$$\mathcal{M}_{\text{low}} = \tilde{\mathbf{L}}\tilde{\mathbf{R}}^{-1}\tilde{\mathbf{L}}, \qquad \mathcal{M}_{\text{high}} = -j\omega^{-1}\tilde{\mathbf{L}}$$

In each case, the preconditioning step is relatively cheap since it does not involve an inner solve. For intermediate frequencies, however, one should use the preconditioner (4.2).

Figure 6 shows that the eigenvalues of the preconditioned reduced system are clustered, almost independent of the filament width h. The system was preconditioned using $\mathcal{M}_{\text{high}}$. The preconditioned system can be solved in few iterations only, and the preconditioner appears to be effective for the mid-frequency range as well. However, for low frequency problems, one should use a two-dimensional Laplace matrix to precondition the real part of the reduced system. It should be noted that accurate inductance extraction may not be needed at low frequencies.

B. Effectiveness of the Preconditioning Approach

There are several advantages of our preconditioning approach. The preconditioning step requires a matrix-vector product that is relatively inexpensive compared to incomplete factorization based preconditioners. The latter involve incomplete factorizations of a partially computed coefficient matrix and triangular solves, which can be expensive, especially on parallel platforms. In addition, experimental evidence suggests that unlike incomplete factorization, our preconditioner is robust and very effective over a wide range of frequencies. To illustrate the effectiveness of the preconditioning scheme, we consider three benchmark problems.

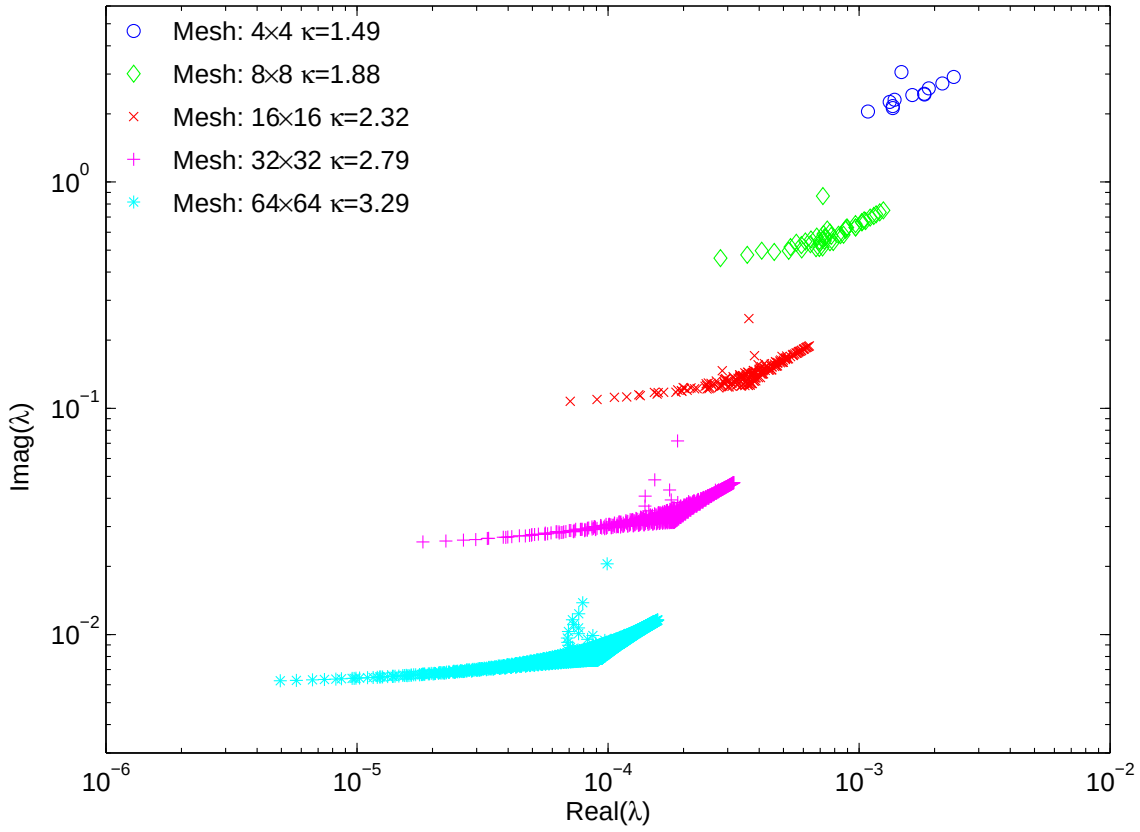

Fig. 6. The spectrum of the preconditioned reduced system for the square ground plane problem. The condition number of the preconditioned system is denoted by κ ($\omega = 2\pi \times$ 10GHz).

1. Ground Plane

The first problem involves computing the self-inductance of a square ground plane of size 1cm $\times$ 1cm (see Fig. 1). The ground plane is used to provide a uniform ground potential to all the components of a VLSI circuit. The plane is discretized by a uniform two-dimensional mesh with mesh width h varying from 2^{-5}cm to 2^{-9}cm. The width of each filament is one-third of its length, and the thickness is 1μm. A tolerance $\tau = 10^{-6}$ was specified on the relative residual norm of GMRES. Table III shows the effectiveness of the preconditioner for the ground plane problem. It can be seen that the number of iterations

Table III. Iterations for convergence of preconditioned GMRES method to compute the self impedance of the ground plane conductor problem. Unpreconditioned GMRES iterations are shown in parenthesis (τ = 10^{-6}).

Mesh Size	Filament Length(cm)	Frequency (f)					
		1 MHz	10 MHz	100 MHz	1GHz	10 GHz	100 GHz
32×32	1/32	26 (75)	24 (68)	11 (36)	8 (30)	8 (30)	8 (30)
64×64	1/64	36 (143)	33 (129)	15 (63)	9 (43)	9 (43)	9 (43)
128×128	1/128	49 (–)	44 (–)	21 (118)	10 (62)	10 (60)	10 (60)
256×256	1/256	65 (–)	58 (–)	29 (–)	12 (94)	11 (83)	11 (83)
512×512	1/512	85 (–)	76 (–)	38 (–)	16 (159)	13 (114)	13 (113)

required by the right-preconditioned GMRES algorithm to solve the linear system (3.3) is almost constant in the high frequency range (1 GHz - 100 GHz) when either the mesh width h or the angular frequency $\omega = 2\pi f$ is changed. The entries marked "–" indicate the inability of iterative solver to reduce the relative residual norm below the threshold set by τ within 200 iterations.

Figures 7 and 8 compare the preconditioning scheme for the ground plane problem with an unpreconditioned GMRES solve. We plot the results for a maximum of 200 iterations and a maximum tolerance of 10^{-8}. It can be seen that over a range of problem discretization, the preconditioning scheme significantly reduces the number of iterations required for convergence. Figure 7 shows that for the high frequency simulations, the growth in iterations across problem discretization is very slow when tolerance is increased, indicating an effective preconditioning scheme. As shown in Fig. 8, the rate of convergence of preconditioned GMRES using $\mathcal{M}_{\text{high}}$ is significantly better than the unpreconditioned approach even for the mid-frequency range.

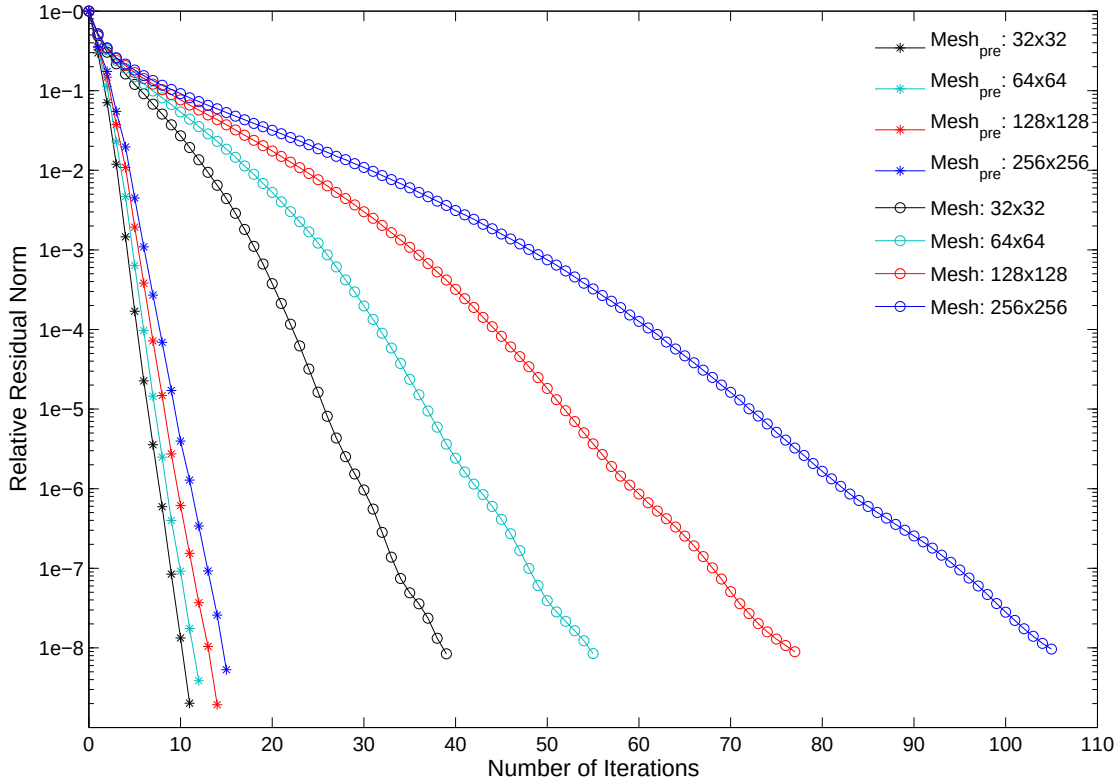

Fig. 7. Comparison of preconditioned system with un-preconditioned system for the ground plane problem at 10GHz.

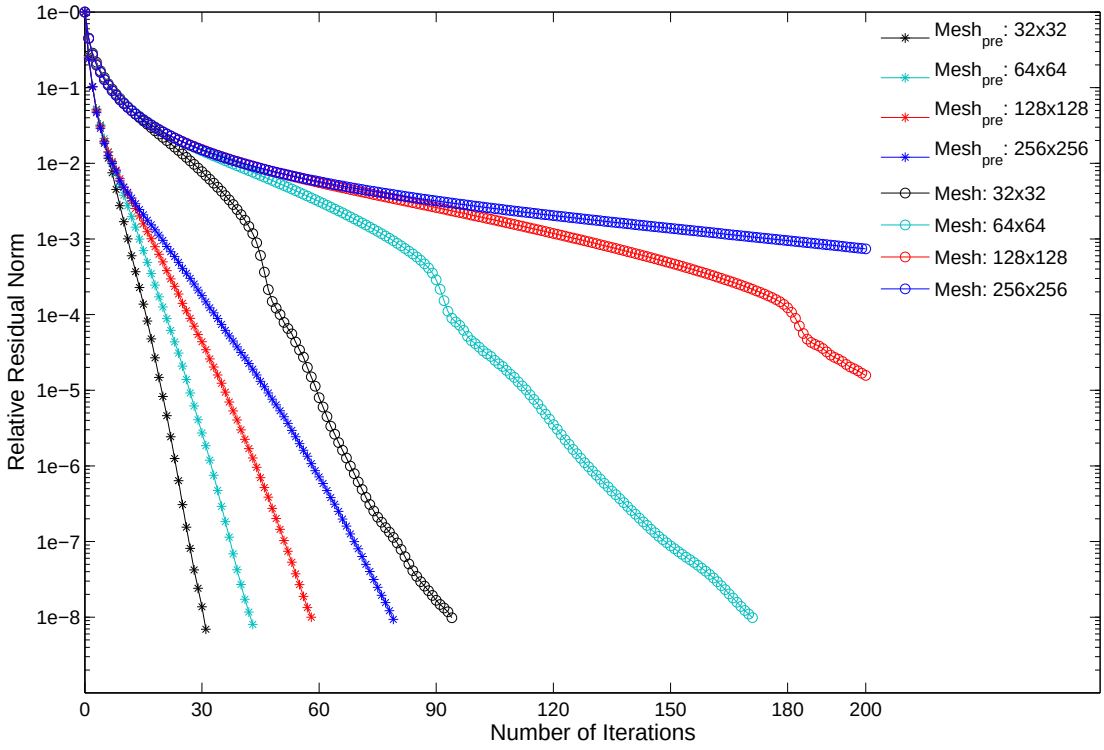

Fig. 8. Comparison of preconditioned system with un-preconditioned system for the ground plane problem at 10MHz.

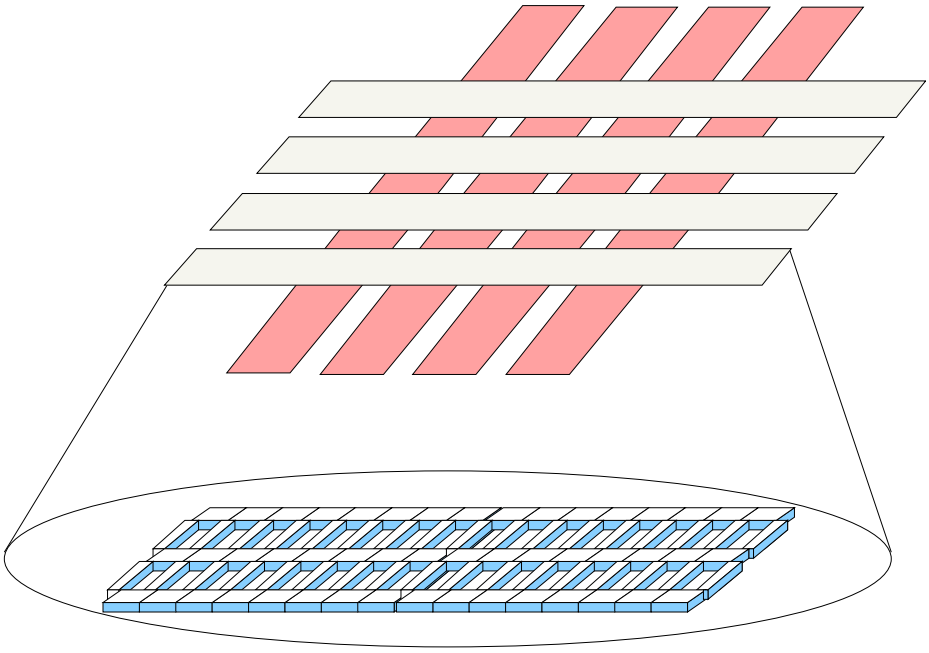

Fig. 9. Cross over problem with a view of a discretized conductor.

2. Cross Over

The second benchmark problem is the *cross over* problem shown in Fig. 9. In VLSI circuits, this is a typical layout configuration with interconnect segments crossing each other on different layers. The problem consists of determining the impedance matrix for these segments. The segments are 2cm long and 2mm wide, and are separated by 300μm in the horizontal direction and by 3mm in vertical direction. The discretization is similar to that of the ground plane problem. These simulations were conducted for a frequency of 10GHz. Tables IV reports the range of iterations required by the right-preconditioned GMRES method to compute the complete impedance matrix, $\mathbf{Z}(\omega)$. For the preconditioned GMRES method, the growth in number of iterations is minimal as the number of conductors in the configuration is increased. Furthermore, the growth in iterations is slow as the mesh is refined. These results illustrate the effectiveness of the preconditioning scheme for typical extraction problems.

Table IV. Iterations for convergence of preconditioned GMRES method for the cross over problem with multiple right-hand sides. Unpreconditioned GMRES iterations are shown in parenthesis ($\tau = 10^{-6}$).

Mesh Size	Filament Length (cm)	Conductor Layout		
		1+1	2+2	4+4
16×160	1/80	11 (34)	13 (37)	15 (38-39)
32×320	1/160	12 (47)	14 (51)	17 (53-54)
64×640	1/320	13 (65)	15-16 (70)	18-19 (73-74)
128×1280	1/640	15 (89-90)	17-18 (95-96)	20-21 (99-102)

3. Pin Connect

The third benchmark problem is the *pin connect* problem shown in Fig. 10. This kind of layout provides connectivity to a chip's pin to various components of the VLSI circuit. The problem consists of determining the complete impedance matrix representing the interaction among the pin structures. This benchmark illustrates the preconditioner's performance for a 3-dimensional problem. We use a two-dimensional discretization of conductor surfaces, similar to that of the ground plane problem. These simulations were conducted for a frequency of 10GHz. Table V reports the range of iterations required by the right-preconditioned GMRES method to compute the complete impedance matrix $\mathbf{Z}(\omega)$, with multiple right-hand sides. For a very coarse mesh discretization, the rate of convergence of the preconditioned GMRES method is weakly dependent on the discretization mesh width h. For thin conductor segments, one dimensional discretization could suffice for the computation of the impedance value.

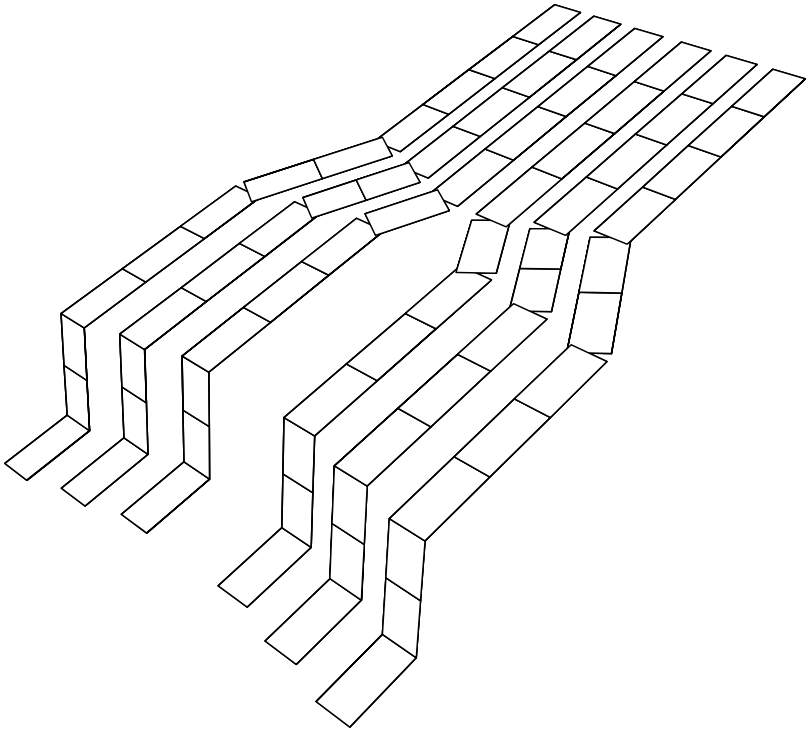

Fig. 10. Pin connect problem with 6 pin.

Table V. Iterations for convergence of preconditioned GMRES method for the pin connect problem with multiple right-hand sides. Unpreconditioned GMRES iterations are shown in parenthesis ($\tau = 10^{-6}$).

Filament Length (cm)	Conductor Layout					
	1 Pin		3 Pin		6 Pin	
	Solenoidal Flows	Iter.	Solenoidal Flows	Iter.	Solenoidal Flows	Iter.
1/20	92	8(14)	270	11(15)	540	12(15)
1/40	368	10(19)	1080	13(20-21)	2160	15(21)
1/80	1472	12(28)	4320	15-16(29-30)	8640	17-18(30-31)
1/160	5888	14(40)	17280	18-19(42-43)	34560	20-21(43)

C. The Inductance Extraction Algorithm

An outline of the preconditioned solenoidal basis method for computing the impedance matrix $\mathbf{Z}(\omega)$ is given below.

Algorithm 1 *Preconditioned Solenoidal Basis Method for Inductance Extraction.*

1. Discretize the surfaces of the conductors with two-dimensional uniform meshes.
2. Compute the solenoidal basis matrix $\mathbf{P}$ for each conductor.
3. For conductor $l = 1, \ldots, n_s$:
   (a) Compute the particular solution $\mathbf{I}_p^{(l)}$ for a unit current flow through conductor l and the corresponding induced potential difference vector $\mathbf{F}^{(l)}$.
   (b) Solve the preconditioned system

$$\mathbf{P}^T \left[\mathbf{R} + j\omega\mathbf{L}\right] \mathbf{P}\tilde{\mathbf{L}}x = \mathbf{P}^T\mathbf{F}^{(l)}, \qquad x^{(l)} = \tilde{\mathbf{L}}x$$

   to determine $x^{(l)}$, and compute filament current and filament potential difference vectors:

$$\mathbf{I}_f^{(l)} = \mathbf{P}x^{(l)} + \mathbf{I}_p^{(l)}, \qquad \mathbf{V}_f^{(l)} = \left[\mathbf{R} + j\omega\mathbf{L}\right] \mathbf{I}_f^{(l)}.$$

   Use right-preconditioned GMRES to solve the system. Use approximate hierarchical methods such as FMM or Barnes-Hut to compute matrix-vector products with $\mathbf{L}$ and $\tilde{\mathbf{L}}$ at each iteration.
   (c) For conductor $k = 1, \ldots, n_s$, determine $\mathbf{Z}(\omega)_{k,l}$ by adding the potential difference across all the filaments along a path between the two ends of conductor k.

An efficient implementation of this algorithm is based on a number of optimizations. The matrix $\mathbf{P}$ is never computed explicitly. A matrix-vector product with $\mathbf{P}$ is used to compute filament currents from mesh currents. Since this computation is defined locally, it can be performed by accumulating the contribution of each mesh current to the four filaments that comprise the mesh or loop. Knowledge of the structure of the discretization mesh is sufficient to develop an implementation in which $\mathbf{P}$ is not computed and stored explicitly. Similarly, matrix-vector products with $\mathbf{P}^T$ are used to compute mesh currents from filament currents. These products can also be computed without explicitly computing $\mathbf{P}^T$. This approach leads to significant saving in storage without increase in computation.

The cost of the orthogonalization step in GMRES is proportional to k^2, where k is the number of iterations. Hence, the parallel performance of GMRES degrades as k increases due to increased communication overhead of orthogonalization step. By using an effective preconditioner that requires very few iterations, we reduce the computational cost as well as the storage requirement of GMRES. Furthermore, the parallel implementation does not suffer from the effects of the orthogonalization step. In the next chapter we demonstrate the numerical and computational superiority of the preconditioner over existing techniques for several benchmark problems.

CHAPTER V

COMPARISON WITH EXISTING WORK

In this chapter, we compare our algorithm with a public domain inductance extraction package called FastHenry [12]. Due to its high accuracy, FastHenry is often used as a reference for all other extraction algorithms. FastHenry uses mesh currents similar to the local solenoidal flows to generate a reduced system. The reduced system is solved by the preconditioned GMRES method to compute accurate estimates of a circuit's parasitic inductance. Matrix-vector products with the dense matrix $\mathbf{L}$ are computed via FMM. The similarities between FastHenry and solenoidal basis method make a compelling case to compare the performance of the two approaches.

The mesh currents in FastHenry are slightly different from the local solenoidal flows in our approach. FastHenry represents the discretized problem as a graph with filaments and external sources as branches in the graph. A mesh current is defined as a current flow along a loop of branches in the graph that does not enclose any other branch. This graph based approach for mesh currents does not exploit the fact that the circular solenoidal flows are discrete curl operators to construct preconditioners.

The main difference between the two algorithms lies in the preconditioning step. FastHenry uses preconditioners that are derived from incomplete factorizations of sparsified forms of the reduced system. These sparse approximations are constructed in a variety of ways. One approach is to use the inverse of blocks of the reduced system. This approach relies on the fact that physically close meshes are tightly coupled. A sparse approximation of the reduced system is obtained by retaining interactions among closely coupled meshes. The preconditioner is then formed by using rows of locally inverted coupling matrix. This approach tends to work well when only a few meshes are close by. Another approach is to use the incomplete LU factors of the reduced system. The preconditioning step is then

implemented as a sequence of forward and backward substitutions.

Other techniques to construct more effective preconditioners for FastHenry include approximation of the dense inductance matrix $\mathbf{L}$ with a sparse matrix $\mathbf{L}_{sp}$ followed by direct factoring of the reduced system. The software allows preconditioners such as DIAG where $\mathbf{L}_{sp}$ = diag($\mathbf{L}$). Other preconditioners such as CUBE and SHELL are also proposed, which explicitly restrict the off-diagonal non-zeros in each column of $\mathbf{L}$ to those resulting from coupling between specific filament pairs. For the SHELL preconditioner, the problem domain is divided into disjoint regions. The sparse approximation $\mathbf{L}_{sp}$ is constructed by placing diagonal blocks of filament interactions in each such region. For the CUBE preconditioner, the 3-dimensional space is divided into cubes and $\mathbf{L}_{sp}$ is composed of blocks of filament interactions in each such cube (see [11] for more details).

Although FastHenry is used as a benchmark for accuracy comparison, it has found limited use in the VLSI-CAD community due to the long simulation time and large memory requirements. Since FastHenry is available only for uniprocessor workstations, the size of problems that can be solved is severely limited. The solenoidal basis method, uses FMM variant to compute products with the system matrix $\mathbf{L}$ and the preconditioner $\tilde{\mathbf{L}}$ directly without explicitly computing these matrices. The resulting implementation is a matrix-free code in which neither the system matrix nor the preconditioner matrix is ever computed. This reduces the storage requirement considerably, thereby allowing larger problems to be solved. Chapter VI provides additional details and outlines an efficient parallel implementation of the solenoidal basis approach.

The performance of the solenoidal basis method was compared to FastHenry on four representative problems: the 2D ground plane problem, the cross over problem in 3D, the pin connect problem and the planar spiral inductor problem. These experiments were conducted on a 1.5 GHz Pentium Workstation with 1 GB of memory running Redhat Linux operating system. Multipoles of degree two were used in all the FMM computations.

Table VI. Comparison of solenoidal basis method and FastHenry for the ground plane problem at 10GHz frequency (Memory in MB and time in seconds).

Mesh	*FASTHENRY-DIAG*			*FASTHENRY-CUBE*			Solenoidal Method		
Size	Iter.	Time	Mem.	Iter.	Time	Mem.	Iter.	Time	Mem.
32×32	28	2.8	12	22	2.8	13	5	1.3	3
64×64	37	18.5	51	32	20.6	55	5	6.1	6
128×128	54	139	219	45	171	233	6	30.2	20
256×256	76	1132	965	63	1596	1036	7	142	78
512×512	–	–	–	–	–	–	7	575	294

A. Ground Plane

Table VI shows the number of iterations needed by preconditioned GMRES to compute the self impedance of a ground plane at 10 GHz (see Fig. 1). A tolerance $\tau = 10^{-3}$ was specified as the stopping criterion for GMRES for both methods. FastHenry was allowed to use default values for all the parameters. In these experiments, the inductance computed by the solenoidal basis method was within 2% of that obtained by FastHenry.

FastHenry requires significant amount of memory to construct the preconditioner matrix and to compute its LU factors. The entries marked "–" indicate the inability of FastHenry to solve the problem within the available system memory. For an unpreconditioned GMRES solve, FastHenry takes 79 iterations for the ground plane problem with a 256×256 conductor refinement. The number of iterations required for convergence of GMRES using DIAG preconditioner was similar to that of the unpreconditioned system. This indicates that the DIAG preconditioning scheme is ineffective for the ground plane problem. Furthermore, a growth in the number of iterations with mesh size indicates a sub-optimal preconditioning scheme that contributes an additional factor towards the increase in cost of

Table VII. Comparison of the preconditioned solenoidal basis method with FastHenry (Memory in MB and time in seconds).

Conductor	*FASTHENRY-DIAG*			*FASTHENRY-CUBE*			Solenoidal Method		
Layout	Iter.	Time	Mem.	Iter.	Time	Mem.	Iter.	Time	Mem.
Mesh size: 32×320									
1	64	73	138	40	65	143	5	16	13
$1+1$	67-73	277	270	43	216	280	6	68	25
$2+2$	74-80	1225	557	44-49	864	574	7-8	344	48
Mesh size: 64×640									
1	89	515	648	58	561	679	6	75	53
$1+1$	–	–	–	–	–	–	7	326	102
$2+2$	–	–	–	–	–	–	8-9	1566	201

solving these systems as the mesh size is increased. Efforts to reduce the storage requirements through greater sparsification tend to decrease the effectiveness of the preconditioner. In contrast, the solenoidal basis method is able to solve the systems in almost fixed number of iterations. One can also restrict memory requirements by fixing the number of Krylov subspace basis vectors in GMRES and using a restarted-GMRES. But, this approach also increases the iterations and solution time.

B. Cross Over

Table VII shows the number of iterations needed by preconditioned GMRES at a frequency of 10GHz for the cross over problem (see Fig. 9). Again, a stopping tolerance $\tau = 10^{-3}$ was used for GMRES for both methods. FastHenry was allowed to use default values for all the parameters. The column marked "Iter." gives the range of iterations needed to solve multiple instances of the linear system. Though FastHenry was able to amortize the

Table VIII. Performance of solenoidal basis method and FastHenry for the pin connect problem at 10GHz frequency (Memory in MB and time in seconds).

No of Pins	No of Filaments	*FASTHENRY-DIAG*			*FASTHENRY-CUBE*			Solenoidal Basis		
		Iter.	Time	Mem.	Iter.	Time	Mem.	Iter.	Time	Mem.
3	8.5K	64-73	103	58	46-52	79	56	8-9	32	7
6	17K	72-81	492	104	49-55	330	100	9-10	159	12
9	26K	76-94	1169	218	50-54	698	209	10-11	411	18

cost of preconditioner construction over calculating multiple columns of $\mathbf{Z}(\omega)$ matrix, the solenoidal basis method still outperforms it. Results show that the reduction in preconditioner construction time and memory when using the DIAG preconditioner are offset by the increased cost of the orthogonalization step in GMRES since a larger number of iterations are required. Additional memory is also required to store the Krylov subspace basis vectors in GMRES. These results also demonstrate the comparative advantage of the solenoidal basis method. The accuracy is similar to the ground plane problem with the inductance value within 3% of that obtained by FastHenry.

C. Pin Connect

Table VIII shows the performance of the solenoidal basis method and FastHenry for the pin connect problem (see Fig. 10). The stopping criterion was the same as the ground plane problem. FastHenry was allowed to use the default CUBE preconditioner. For thin segments, FastHenry primarily discretizes the conductor surface using long filaments along the width only. The number of filaments and meshes obtained by such an approach would be similar. On the other hand, solenoidal basis method uses two-dimensional approach that yields nearly twice the number of filaments compared to the number of meshes. We report the performance of the two approaches with a conductor surface discretization that yields

similar number of filaments. Results in Table VIII show that with increase in number of filaments, the growth in the number of iterations for the solenoidal basis method is minimal. Although the cost of preconditioner construction for FastHenry is significantly less for this problem, the increase in iterations results in a higher solution time. One can expect further increase in time and memory requirements for FastHenry as the problem size is increased.

In these experiments, the inductance computed by the solenoidal basis method was within 4% of that obtained by FastHenry. The difference in the inductance value between the two appraoches is also due to the difference in modelling of the angular joints within each pin. FastHenry uses overlapping regions at the turns, whereas solenoidal method uses non-overlapping regions. One should note that for FastHenry the one dimensional discretization scheme requires less time and memory as compared to a two dimensional discretization, irrespective of the preconditioning approach.

D. Spiral Inductor

Another benchmark problem that we use to compare the performance of solenoidal method with FastHenry is the *spiral-inductor* problem shown in Fig. 11. The spiral inductor problem is a challenging example consisting of a coil shaped conductor which is used in electromagnetic circuitry such as in magnetic access cards. The problem consists of determining the self-impedance of the structure. The segments are 1mm wide, and are separated by approximately 1mm. The discretization is similar to that of the ground plane problem. The simulations were conducted for a frequency of 10GHz.

Table IX compares the performance of the solenoidal basis method with FastHenry. The growth in the number of iterations for the solenoidal basis method is minimal due to the preconditioning scheme. For FastHenry, the preconditioner construction cost for spiral-inductor is similar to that of ground plane problem. The growth in time and memory

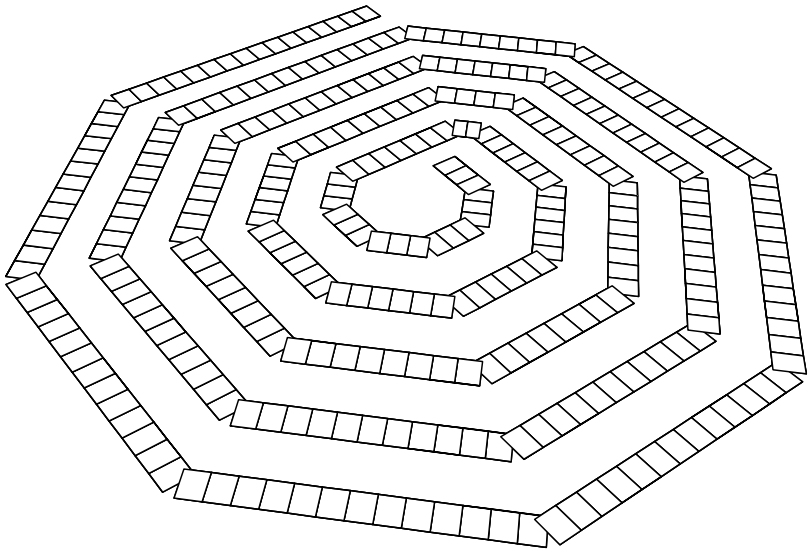

Fig. 11. Spiral inductor.

requirement with increase in problem size further ascertain this. To do a fair comparison of the the two approaches, we force FastHenry to generate a discretization with similar number of filaments as used by the solenoidal method. The 3 rows of the Table IX corresponds to a discretization of 7, 9 and 13 meshes along the width of each spiral turn for the solenoidal method and 9, 7 and 15 meshes for FastHenry. In these experiments, the inductance computed by the solenoidal basis method was within 6% of that obtained by FastHenry. The additional difference in the inductance value between two approaches is

Table IX. Comparison of solenoidal basis method and FastHenry for the spiral inductor problem at 10GHz frequency (Memory in MB and time in seconds).

No of	*FASTHENRY-DIAG*			*FASTHENRY-CUBE*			Solenoidal Method		
Filaments	Iter.	Time	Mem.	Iter.	Time	Mem.	Iter.	Time	Mem.
24K	55	76	132	37	94	131	10	35	17
40K	66	151	201	42	191	205	10	59	26
80K	81	312	464	53	648	457	11	125	49

due to the modelling difference of the turns of the spiral, and does not necessarily indicate lower accuracy.

The modest performance of the preconditioners in FastHenry come with a significant cost of computing the preconditioners as well as storing them. While these storage requirements can be reduced by computing incomplete factorizations, often this results in a weak preconditioner. The slower convergence rates associated with ineffective preconditioning may lead to overall higher computational cost. The comparative advantage of the solenoidal method is expected to grow with larger problems.

The performance of the solenoidal method can be further boosted by using additional memory. It is possible to store direct interactions during the initial matrix-vector product with $\mathbf{L}$ and $\tilde{\mathbf{L}}$, and to reuse them later. This approach is beneficial when there are multiple right hand sides.

CHAPTER VI

PARALLEL FORMULATION*

The most computationally intensive steps in the algorithm are the matrix-vector products with the reduced system matrix $\mathbf{P}^T [\mathbf{R} + j\omega\mathbf{L}] \mathbf{P}$ and the preconditioner matrix $\mathcal{M}$. The cost of multiplying a vector with the dense matrix $\mathbf{L}$ is significantly greater than the multiplication with $\mathbf{P}$ or $\mathbf{P}^T$. It is worthwhile to use multipole-based hierarchical methods to compute matrix-vector products with $\mathbf{L}$. Furthermore, since the structure of the preconditioner $\tilde{\mathbf{L}}$ is similar to $\mathbf{L}$, these fast methods should be used to compute matrix-vector products in the preconditioning step as well. A number of such techniques have been developed including the Appel's algorithm [1], Barnes-Hut [3] method and the well known Fast Multipole Method (FMM) [7]. Other vector operations in GMRES can be parallelized in a straight-forward manner. The implementations based on the hierarchical methods are *matrix-free* approaches in which neither the system matrix nor the preconditioner matrix is ever computed. This reduces the storage requirement considerably, thereby allowing larger problems to be solved.

A. Hierarchical Dense Matrix-Vector Products

The cost of computing an accurate matrix-vector products with a dense $n \times n$ matrix require $O(n^2)$ operations. If the entries of the dense matrix have a $1/r$ decaying kernel, approximations to these products can be computed efficiently through hierarchical multipole-based methods. The nature of the elements in $\mathbf{L}$ and $\tilde{\mathbf{L}}$ allows use of fast hierarchical algorithms in which reduction in computational complexity is obtained at the expense of accuracy. In par-

---

*Part of this chapter is reprinted from Parallel Computing, Vol. 29, by H. Mahawar and V. Sarin, "Parallel Iterative Methods for Dense Linear Systems in Inductance Extraction", 1219-1235., Copyright (2003), with permission from Elsevier.

ticular, one can exploit the rapid decay of the kernel with distance to compute approximate matrix-vector products in $O(n \log n)$ or $O(n)$ operations. For the inductance extraction problem, these algorithms use a truncated series to approximate the effect of a cluster of filament currents on other clusters that are well-separated. The Barnes-Hut method relies only on filament-cluster interactions to achieve an $O(n \log n)$ computational bound whereas the FMM uses both filament-cluster and cluster-cluster interactions to achieve an $O(n)$ bound for uniform filament distributions. The accuracy of FMM can be improved by increasing the multipole degree d, which determines the number of terms used in the approximation. A hierarchical multipole method (HMM) [18] has also been developed. The HMM implementation can be treated as either augmented Barnes-Hut or modified FMM, and has the implementation advantages of Barnes-Hut and accuracy similar to FMM. Similar to Barnes-Hut, HMM relies only on filament-cluster interactions to achieve an $O(n \log n)$ computational bound. In this work, we use a variant of FMM algorithm to compute approximate matrix-vector products with both $\mathbf{L}$ and $\tilde{\mathbf{L}}$. These matrices compute potential at each filament due to current flow in all the filaments. Filament mid-points form the set of particles for FMM. It is sufficient to represent the filaments by their mid-points to compute the inductive effect between a pair of filaments. Self-inductance of a filament is computed by closed form formula [9].

These hierarchical algorithms work in two phases: the oct-tree construction phase and the potential evaluation phase. 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 non-overlapping subdomains. The points are also partitioned among the subdomains. The process is repeated on each subdomain recursively until every subdomain has at most s filaments, where s is a parameter chosen to maximize computational efficiency. This recursive strategy yields an oct-tree with a hierarchical spatial ordering of the points.

A subdomain is represented by a subtree whose leaf nodes contain the filaments in the subdomain. Since the oct-tree stores non-empty cubes only, this scheme yields non-uniform oct-trees for unstructured point distributions. During the potential evaluation phase, each internal node in the tree computes and stores the effect of all the particles contained in its sub-tree. The effect at a node is computed from the effect of its children through an up-traversal of the nodes from the leaves to the root. To compute the effect of remaining particles that belong to other subtrees, each internal node uses the information stored at the roots of other subtrees that are well-separated. Well-separateness is established by using a distance metric to determine if the nodes are sufficiently "far-off" such that the error in accuracy is below a threshold. Using a top-down scheme, the accumulated effect at the root of a subtree is passed down to the leaf nodes. The effect of nearby filaments are calculated directly.

During potential evaluation phase of Barnes-Hut method, one calculates the *center of mass* at the internal nodes in a bottom-up fashion. The center of mass at each node approximates the effect of all particles in its subtree. To compute the effect due to the node's particles at an observation point, a top-down traversal is done to identify nodes that satisfy the multipole acceptance criteria. The acceptance criteria requires that the ratio of the node's size to the distance between the node's center and the observation point be less than a threshold value. To get the effect of the particles in a node's subtree, one has to use the center of mass of the node. More details on it can be found in [3].

The potential evaluation phase of FMM consists of two traversals of the tree. For each node, FMM computes a set of multipole coefficients in a bottom-up fashion. These coefficients can be used to compute the potential due to all the filaments within the node's subdomain at an observation point outside the subdomain. The observation point must be outside a sphere that encloses the subdomain completely. For simplicity, a larger cube containing the sphere can be chosen as the neighborhood of a subdomain. The computational

complexity of this step is $O((d+1)^4 n)$ for a problem with n points (see [7] 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. These local coefficients can be used to compute the potential at any point inside a node's subdomain due to 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.

1. Impact of Parameters

For FMM, when using d degree multipoles, the number of multipole and local coefficients at each node is $(d+1)^2$ and the cost of computing these coefficients is proportional to $(d+1)^4$. Increase in multipole degree has dual benefits. In addition to the increase in accuracy of the approximation, the parallel performance improves significantly due to rapid growth in the computation. The speedup often exhibits superlinear behavior due to the cache-friendly nature of these computations. Tables X and XI show the effect of increasing multipole degree on the ground plane and cross over interconnects problems, respectively. These experiments were conducted on a 128-processor SGI Origin2000 with 250MHz clock speed at the National Center for Supercomputer Applications (NCSA) at the University of Illinois. OpenMP directives were used to parallelize the code. Figures 12 and 13 show that for a fixed problem size, the parallel efficiency increases with multipole degree for a fixed number of processor (see [15] for more details).

Table X. Parallel performance of the ground plane problem for different choices of multipole degree (s=32, conductor mesh size=256×256, time in seconds).

Multipole degree (d)	No. of processors (p)					
	p = 16		p = 32		p = 64	
	Time	Speedup	Time	Speedup	Time	Speedup
1	36	9.2	21	16.1	17	20.2
2	91	13.6	51	24.1	30	40.8
4	604	14.7	320	27.8	167	53.4
6	2298	18.1	1195	34.7	642	64.5

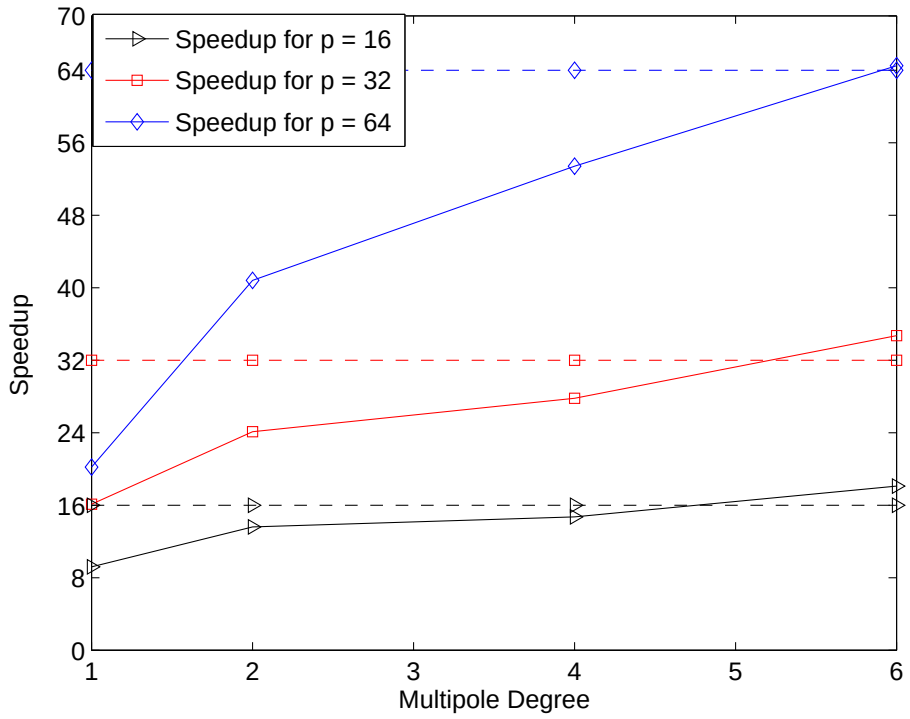

Fig. 12. Effect of multipole degree on the ground plane problem. The dashed lines indicate maximum theoretical speedup on p processors.

Table XI. Parallel performance of the overlapping panels problem for different choices of multipole degree (s=32, conductor mesh size=64×256, time in seconds).

Multipole degree (d)	No. of processors (p)					
	p = 16		p = 32		p = 64	
	Time	Speedup	Time	Speedup	Time	Speedup
1	42	9.4	22	17.8	20	19.5
2	100	13.1	55	23.3	33	39.2
4	757	15.0	395	28.8	175	64.9
6	2325	18.7	1213	35.8	642	67.6

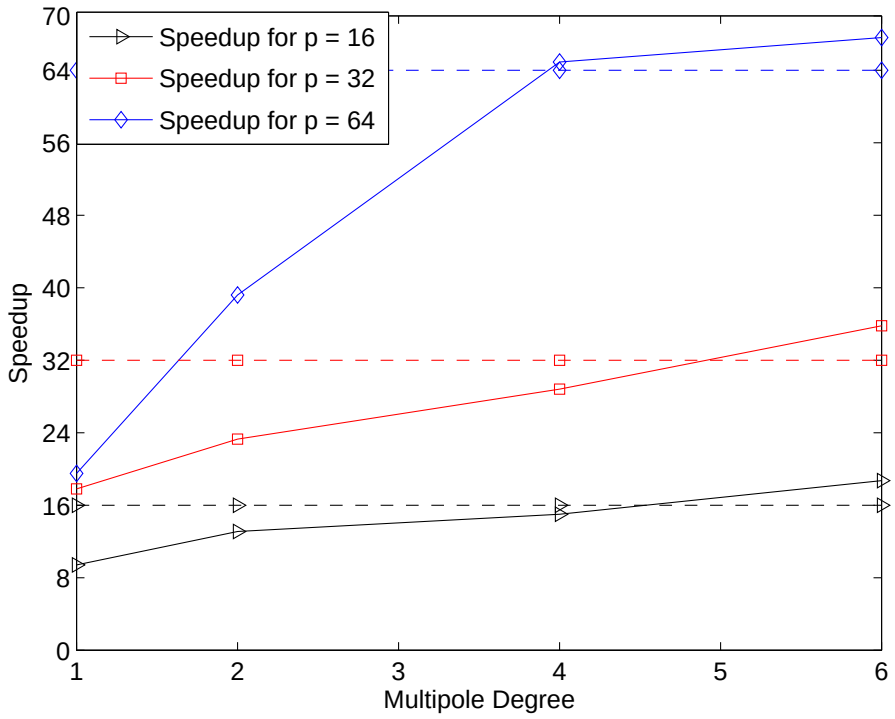

Fig. 13. Effect of multipole degree on the cross over problem. The dashed lines indicate the maximum theoretical speedup on p processors.

Table XII. Effect of the multipole degree on the serial execution time for different choices of maximum particles per leaf box (conductor mesh size =32×32, time in seconds).

Multipole degree (d)	Particles per leaf box (s)			
	s = 2	s = 8	s = 32	s = 128
1	50	18	13	30
2	226	63	25	33
4	1513	398	111	51

The performance of the hierarchical multipole algorithms also depends on the maximum number of particles allowed per leaf box (s). The dominant computation in FMM consists of multipole-to-local translations (M2L) with computational cost proportional to $(d+1)^4$. Table XII shows that with increase in d, the FMM time increases proportional to $(d+1)^4$. The execution time for FMM decreases when s is increased due to a decrease in the number of M2Ls. The cost of direct interactions is proportional to s^2 and is negligible for small values of s. Direct interactions begin to dominate the overall cost for large values of s, resulting in higher execution time. These experiments were conducted on a 64-bit AMD Opteron workstation with a 1.4GHz processor running SuSE-Linux operating system. Table XII shows that when s is increased, the FMM execution time reduces rapidly due to reduction in M2Ls, until the direct interactions begin to dominate the computational cost. For a given problem, one can identify (d, s) pair that minimizes the execution time (see [18] for more details).

B. ParIS - Parallel Inductance Extraction Software

We have developed an object-oriented parallel implementation of the solenoidal basis algorithm for inductance extraction. This software combines the advantages of the solenoidal basis method, fast hierarchical methods for dense matrix-vector products, and a highly ef-

fective preconditioning scheme to provide a powerful package for inductance extraction. In addition, the software includes an efficient parallel implementation that reduces the overall computation time on parallel architectures [16, 17, 19].

The building blocks of *ParIS* are conductor elements. Each conductor is uniformly discretized with a mesh of filaments. To exploit parallelism at the conductor level, each conductor is assigned to a different processor. All the data structures that are native to a conductor are local to its processor. This includes the filaments in a conductor and the associated FMM tree. With the exception of matrix-vector products, all other computations are local to each conductor. Only matrix-vector products incur communication cost as they involve interactions among different conductors that are distributed across processors.

The matrix-vector product with the inductance matrix $\mathbf{L}$ and the preconditioner $\tilde{\mathbf{L}}$ involve interactions among filaments of the same conductor as well as between the filaments of different conductors. Interactions between the filaments of the same conductor are computed locally by the associated processor. To get the effect of filaments in other conductors, a processor needs to exchange multipole coefficients with other processors. During a preprocessing step, *ParIS* identifies the nodes in a conductor's tree that are required by other conductors. The cost of this step is amortized over the iterations of the solver. While computing the dense matrix-vector product, communication is needed to translate the multipole coefficients of these nodes to the nodes on other processors. Communication is also needed when computing direct interactions between adjacent nodes that belong to different subtrees. This type of communication is proportional to the number of filaments on the subdomain boundary.

A straightforward approach for parallelization of matrix-vector product can be implemented by constructing a single oct-tree with filament mid-points as set of particles. Such a tree structure can easily identify the nodes that participate in the communication phases without significant overheads. At every level of the hierarchical oct-tree, nodes should be

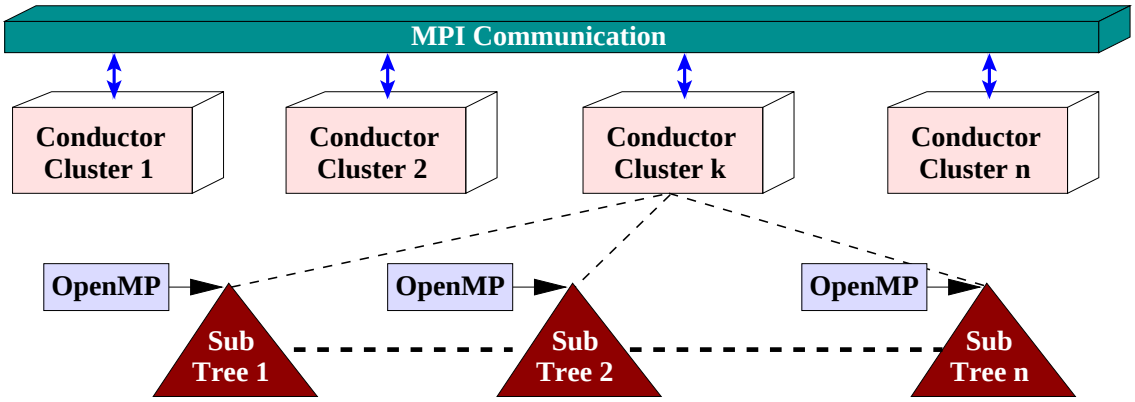

Fig. 14. Two-tier parallelization scheme implemented in *ParIS*

assigned to the processors such that the computational work is evenly distributed. This can be achieved by using a cost function that accounts for cost of all interactions for a given node. To reduce communication costs, particles should be partitioned across processors in such a way that the boundary points on the processor subdomain are contiguous in memory locations. This can be achieved by various space filling curves such as Morton ordering, Hilbert ordering, Gray code, etc. [2]. A parallel FMM implementation using such a scheme for capacitance extraction problem has been presented in [30]. Various parallel formulations of multipole-based techniques have been developed by several groups [6, 24, 27, 29].

Additional parallelism is available within each conductor through the FMM structure. Partitioning of the oct-tree of a conductor among multiple threads is achieved by assigning non-overlapping subdomains at a particular level k of the oct-tree. The internal nodes at level k are roots of subtrees of the corresponding subdomains. With this partitioning, a thread is responsible for calculating the multipole and local coefficients of its own sub-tree. The computation involving these coefficients requires no communication between threads. The computation for the levels above k can be be assigned to small number of threads to increase the parallel efficiency. With different sized conductors, one can have more threads associated with larger conductors. This scheme allows load balancing to a certain extent.

A two-tier parallelization approach shown in Fig. 14 allows the algorithm to be implemented in hybrid or mixed mode using both MPI and OpenMP directives. The software can be executed on a variety of platforms ranging from shared-memory multiprocessors to workstation clusters seamlessly.

C. Parallel Performance

The software design of *ParIS* has the dual advantage of portability and performance across a variety of platforms. This is achieved through a two-tier parallelization approach that uses MPI processes for conductor level parallelism, while OpenMP directives are used to exploit parallelism within a conductor. We present experiments to demonstrate the parallel performance of the software on multiprocessors with shared, distributed, and distributed-shared memory architectures. We consider distributed memory platforms such as the 64-bit AMD Linux cluster where parallelism can be exploited via MPI processes only. Distributed-shared memory platforms such as the IBM p690 that allow mixed mode parallelization with both MPI and OpenMP are also considered.

We present numerical experiments to study the parallel performance of the software. These experiments were designed to illustrate the parallel efficiency and scalability of the implementation for benchmark problems. We report the execution time and parallel efficiency of the iterative solver. Efficiency is defined as the percent utilization of the processors, and is computed as the ratio of speedup to the number of processors used. Speedup refers to the speed improvement obtained by the parallel code over a single processor execution.

Instead of solving the full inductance extraction problem, we report the parallel performance of the algorithm for a fixed number of GMRES iterations. Each iteration involved dense matrix-vector products with the coefficient matrix as well as the preconditioner. This

is indicative of the actual performance since the dense matrix-vector products account for over 98% of the execution time. The performance of the software depends on various parameters for FMM, such as the number of particles in leaf nodes (s), the multipole degree (d), etc. One should note that the higher multipole degree significantly increases the computional cost compared to the communication cost, which in turn improves the parallel effficiency (see [15] for details).

1. Shared Memory Parallelization

We use the ground plane problem shown in Fig. 1 to illustrate the parallel performance of the code on a shared memory multiprocessor. These experiments were conducted on a 32-processor IBM p690 multiprocessor with 1.3GHz processor speed and AIX5.1 operating system. The code was parallelized with OpenMP directives only.

Table XIII shows the execution time and parallel efficiency of the software for linear systems of order 32K, 128K and 512K unknowns. For a fixed size problem, a modest decrease in parallel efficiency with increase in the number of processors indicates an efficient parallel implementation. This effect is pronounced due to the increase in the serial component of the matrix-vector routine corresponding to the top k levels of the oct-tree. Figure 15 illustrates the scalability of the algorithm. It can be seen that by increasing the problem size, parallel efficiency is maintained when the number of processors are increased.

Table XIII. Parallel performance for the ground plane problem on IBM p690 using OpenMP (d=4, s=32, time in seconds).

No. of processors	Mesh Size					
	128 × 128		256 × 256		512 × 512	
	Time	%Eff.	Time	%Eff.	Time	%Eff.
1	60.4	100	260	100	1063	100
2	31.3	97	133	98	546	97
4	15.6	97	67	97	276	96
8	8.9	85	37	88	148	90
16	5.5	68	22	73	93	72

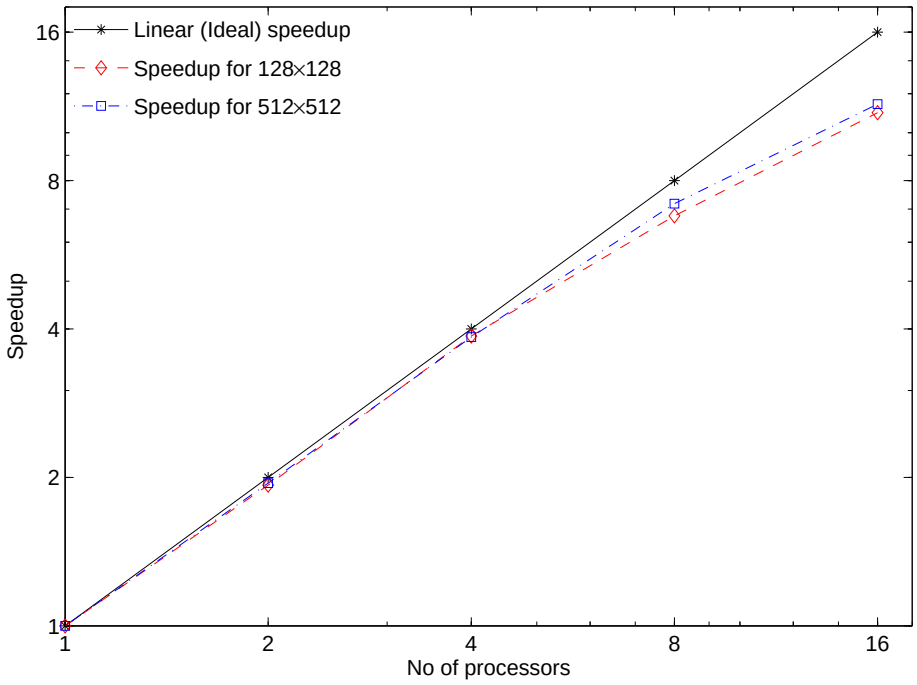

Fig. 15. Shared memory speedup for the ground plane problem with different conductor discretization.

Table XIV. Parallel performance for the cross over problem on IBM p690 using MPI and OpenMP (conductor mesh size=32×320, d=4, s=64, time in seconds).

	P_{MPI}= 1		P_{MPI}= 2		P_{MPI}= 4		P_{MPI}= 8	
P_{OMP}	Time	%Eff	Time	%Eff	Time	%Eff	Time	%Eff
1	8012	100	4091	98	2053	98	1119	90
2	4033	99	2028	99	1070	94	598	84
4	2118	95	1098	91	585	86		
8	1452	69	753	67				

2. Mixed Mode Parallelization

The benchmark problem presented in this section utilizes the two-tier parallel implementation of the software. The cross over problem shown in Fig. 9 is a standard benchmark problem for inductance extraction. The problem consists of determining the impedance matrix of 16 overlapping segments in a three-dimensional configuration. These 16 conductors were spread out in 2 layers of 8 conductors each in 3-dimension. This problem leads to a non-uniform point distribution for the dense matrix-vector multiplication algorithm. Mixed mode experiments were conducted on 16 processors of an IBM p690 at NCSA, Illinois. No more than 16 processors were available due to site restrictions. Various combinations of OpenMP (P_{OMP}) and MPI processes (P_{MPI}) were used to demonstrate the mixed mode parallel performance of the software. MPI processes were assigned conductors and OpenMP directives were used to parallelize computation within conductors.

Table XIV shows the parallel performance of the software where each conductor has been discretized by a mesh of size 32 $\times$ 320. The experiments were setup to compute the full impedance matrix. For the given problem size, the linear system includes 320K unknowns. The speedup obtained by the code resembles the ground plane problem, showing that parallel performance of the code does not diminish for three-dimensional problems.

Table XV. Parallel performance for the 8+8 cross over problem using MPI on IBM p690 and AMD-64 Linux cluster (d=4, s=64, time in seconds).

No of processors	IBM p690		AMD-64 Linux	
	Time	%Eff.	Time	%Eff.
Conductor Mesh Size: 32×320				
1	8305	100	16270	100
2	4151	100	9183	89
4	2085	99	5019	81
8	1199	87	2436	84
16	659	79	1408	72
Conductor Mesh Size: 64×640				
1	31656	100	63890	100
2	15821	100	34445	93
4	7955	99	18877	85
8	4401	90	9360	85
16	2370	84	5168	77

3. Distributed Memory Parallelization

The preconditioned iterative solver outlined earlier can be implemented efficiently on distributed-memory multiprocessors. The experiments reported in this section were conducted on IBM p690 at NCSA and a 64-bit AMD Opteron-240 Tensor cluster at Texas A&M University. The Tensor cluster consists of 1.4GHz 64-bit AMD Opteron processors with SuSE-Linux operating system. Portland Group (PGI) compilers were used on the Tensor cluster for compiling the code.

Table XV shows the execution time and parallel performance of the software for the cross over problem with 16 conductors. The parallel implementation uses MPI directives

only. For the two problem instances with variable conductor discretization, the linear system includes 320K and 1280K unknowns. The parallel performance of the software is nearly identical on both multiprocessors. This indicates that the code utilizes each processor efficiently on both systems when the load is distributed uniformly across processes. Note that the drop in parallel performance is due to increase in communication as well as computations.

The parallel performance of the algorithm is also analyzed by conducting experiments with larger number of processors. Table XVI shows the parallel performance of the software for the 4-layered cross over problem with 128 conductors on up to 128 processors of Tensor cluster. These 128 conductors were spread out in 4 layers of 32 conductors each in 3-dimension. Note that we only solve for first 16 columns of the impedance matrix. For the two problem instance with variable conductor discretization, the linear system includes 2560K and 10240K unknowns. Due to memory restrictions, it was not feasible to run larger problem instances on fewer than 16 processors. Figure 16 reports the speedup over 16 processor run for all problem instances. The results demonstrate that the software is able to maintain high parallel performance on large number of processors.

The performance of the software can also be compared in terms of the processor utilization of the code. This measure of parallel performance is especially useful to compare problems that require different number of mutual inductance interactions (see [17] for details).

Table XVI. Parallel performance for the 4-layered cross over problem using MPI on AMD-64 Cluster (d=4, s=32, time in seconds).

No of processors	Conductor Discretization			
	32×320		64×640	
	Time	%Eff.	Time	%Eff.
16	19812	100	81182	100
32	10071	98	41685	97
64	5050	98	21013	97
128	3163	78	10841	94

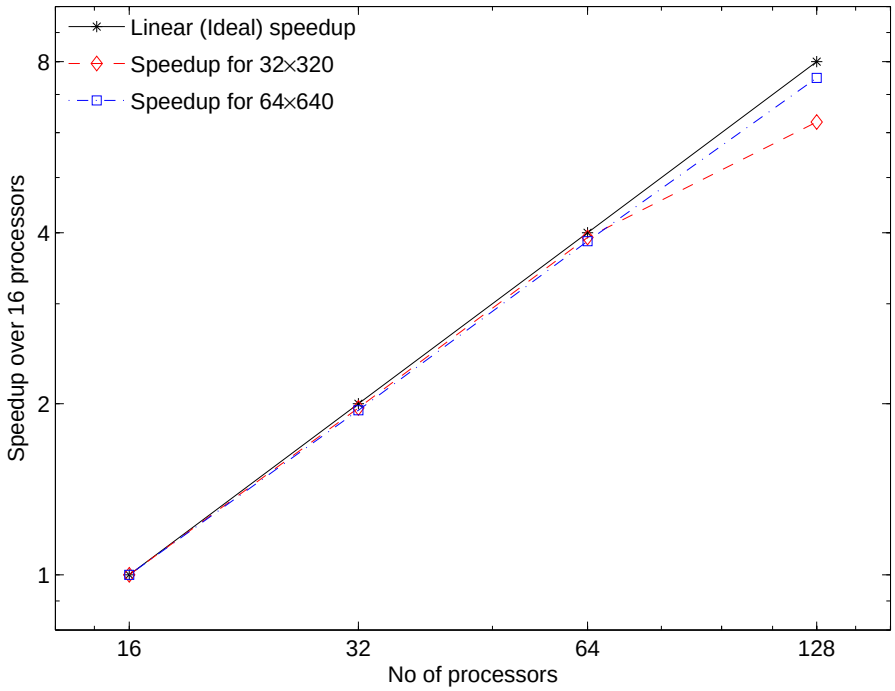

Fig. 16. Parallel performance of the software for 4-layered cross over problem on up to 128 processor of Tensor cluster.

CHAPTER VII

CONCLUSIONS

For the modern VLSI circuits, with advances in circuit technology and increasing operational frequency, there is a significant need for fast and accurate inductance extraction software. The computation of impedance matrix for the inductance extraction problem requires solution of dense, complex linear systems of equations.

This work presents a high performance parallel preconditioned software for inductance extraction of VLSI circuits. The software derives its strength from an effective preconditioning scheme that results in rapid convergence of the iterative method. We use local solenoidal flows to convert the system into a reduced system that is solved iteratively by the preconditioned GMRES method. Fast hierarchical multipole-based methods were used for the computationally intensive matrix-vector products with the dense coefficient and preconditioner matrices. We have proposed a nearly optimal preconditioner based on the inductive coupling among filaments of the mesh that does not require explicit construction of the coefficient matrix. Numerical experiments indicate that for high frequency extraction in the range of 100MHz - 100GHz, the proposed preconditioner exhibits convergence in almost fixed number of iterations irrespective of the mesh refinement and operational frequency. Benchmark experiments indicate that the serial performance of the algorithm is superior to FastHenry, a well-known inductance extraction package.

Furthermore, we present an efficient parallel implementation of the preconditioned algorithm that reduces the overall computation time considerably on multiprocessors. The software employs a two-tier parallelization approach involving MPI and OpenMP directives to deliver a high performance parallel software that is portable to a variety of multiprocessors. Experimental results demonstrate that the parallel implementation achieves very high parallel efficiency on shared-memory, distributed-memory, and distributed-shared mem-

ory multiprocessors.

A. Possible Enhancements

There are a number of enhancements that are possible:

- The two-dimensional uniform discretization of conductor surfaces yields spatially uniform particle distribution for the hierarchical methods. An adaptive refinement approach for conductor surfaces, one that is similar to capacitance extraction [26], can be used.

- Solenoidal basis method gives a reduced system of equations that only depend on unknown mesh currents. One can develop fast hierarchical methods for the $1/r^3$ kernel, which is directly applicable to the reduced system. The mesh based formulation is same as applying discrete curl operator to $1/r$ kernel. An approach using Gegenbauer polynomials can be used to generalize the multipole based hierarchical scheme to compute fast approximate matrix-vector product with $r^{-\lambda}$ kernel [28].

- An alternate parallelization scheme bases on a single oct-tree can be used, which utilizes the particle distribution rather than conductor boundaries. This would result in a load-balanced parallel implementation for the matrix-vector products.

REFERENCES

[1] A. APPEL, *An efficient program for many-body simulation*, SIAM Journal on Scientific and Statistical Computing, 6 (1985), pp. 85–103.

[2] T. ASANO, D. RANJAN, T. ROOS, E. WELZL, AND P. WIDMAYER, *Space-filling curves and their use in the design of geometric data structures*, Theoretical Computer Science, 181 (1997), pp. 3–15.

[3] J. BARNES AND P. HUT, *A hierarchical $O(n\ log\ n)$ force calculation algorithm*, Nature, 324 (1986), pp. 446–449.

[4] A. C. CANGELLARIS, J. L. PRINCE, AND L. P. VAKANAS, *Frequency-dependent inductance and resistance calculation for three-dimensional structures in high-speed interconnect systems*, IEEE Transactions on Components, Hybrids, and Manufacturing Technology, 13 (1990), pp. 154–159.

[5] K. GALA, V. ZOLOTOV, R. PANDA, B. YOUNG, J. WANG, AND D. BLAAUW, *On-chip inductance modeling and analysis*, in Proceedings of the 37^{th} Design Automation Conference, Los Angeles, California, June 2000, pp. 63–68.

[6] A. GRAMA, V. KUMAR, AND A. SAMEH, *Parallel hierarchical solvers and preconditioners for boundary element methods.*, SIAM Journal of Scientific Computing, 20 (1998), pp. 337–358.

[7] L. GREENGARD, *The Rapid Evaluation of Potential Fields in Particle Systems*, The MIT Press, Cambridge, Massachusetts, 1988.

[8] L. GREENGARD AND V. ROKHLIN, *A fast algorithm for particle simulations*, Journal of Computational Physics, 73 (1987), pp. 325–348.

[9] F. W. GROVER, *Inductance Calculations, Working Formulas and Tables*, D. Van Nostrand Company, Inc, New York, 1947.

[10] Z. HE, M. CELIK, AND L. PILEGGI, *Spie: Sparse partial inductance extraction*, in Proceedings of the 34^{th} Design Automation Conference, Anaheim, California, June 1997, pp. 137–140.

[11] M. KAMON, *Fast Parasatic Extraction and Simultaion of Three-dimensional Interconnect via Quasistatic Analysis*, PhD dissertation, Massachusetts Institute of Technology, February 1998.

[12] M. KAMON, M. J. TSUK, AND J. WHITE, *FASTHENRY: A multipole-accelerated 3D inductance extraction program*, IEEE Transaction on Microwave Theory and Techniques, 42 (1994), pp. 1750–1758.

[13] B. KRAUTER AND S. MEHROTRA, *Layout based frequency dependent inductance and resistance extraction for on-chip interconnect timing analysis*, in Proceedings of the 35^{th} Design Automation Conference, San Francisco, California, June 1998, pp. 303–308.

[14] B. KRAUTER AND L. T. PILEGGI, *Generating sparse partial inductance matrices with guaranteed stability*, in Proceedings of the International Conference on Computer-Aided Design, San Jose, California, November 1995, pp. 45–52.

[15] H. MAHAWAR AND V. SARIN, *Parallel iterative methods for dense linear systems in inductance extraction*, Parallel Computing, 29 (2003), pp. 1219–1235.

[16] H. MAHAWAR AND V. SARIN, *Parallel software for inductance extraction*, in Proceedings of the 33^{rd} International Conference on Parallel Processing, Quebec, Canada, August 2004, pp. 380–386.

[17] H. MAHAWAR AND V. SARIN, *Parallel algorithms for inductance extraction of vlsi circuits*, in Proceedings of the 20^{th} International Parallel and Distributed Processing Symposium, Rhodes Island, Greece (To appear), April 2006.

[18] H. MAHAWAR, V. SARIN, AND A. GRAMA, *Parallel performance of hierarchical multipole algorithms for inductance extraction*, in Proceedings of the 11^{th} International Conference on High Performance Computing, vol. 3296 of Lecture Notes in Computer Science, Springer-Verlag, Bangalore, India, December 2004, pp. 450–461.

[19] H. MAHAWAR, V. SARIN, AND W. SHI, *Fast inductance extraction of large vlsi circuits*, in Proceedings of the 16^{th} International Parallel and Distributed Processing Symposium, Fort Lauderdale, Florida, April 2002.

[20] H. MAHAWAR, V. SARIN, AND W. SHI, *A solenoidal basis method for efficient inductance extraction*, in Proceedings of the 39^{th} Design Automation Conference, New Orleans, Louisiana, June 2002, pp. 751–756.

[21] A. E. RUEHLI, *Inductance calculations in a complex integrated circuit environment*, IBM Journal of Research and Development, 16 (1972), pp. 470–481.

[22] Y. SAAD, *Iterative Methods for Sparse Linear Systems*, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, 2003.

[23] S. R. SAMBAVARAM AND V. SARIN, *A parallel solenoidal basis method for incompressible fluid flow problems*, in Proceedings of the 13^{th} International Conference on Parallel Computational Fluid Dynamics, P. Wilders, ed., Egmond aan Zee, Amsterdam, May 2001, pp. 309–314.

[24] F. SEVILGEN, S. ALURU, AND N. FUTAMURA, *A provably optimal, distribution-independent, parallel fast multipole method*, in Proceedings of the 14^{th} IEEE Interna-

tional Parallel and Distributed Processing Symposium, Cancun, Mexico, May 2000, pp. 77–84.

[25] K. L. SHEPARD AND Z. TIAN, *Return-limited inductances: A practical approach to on-chip inductance extraction*, IEEE Transaction on Computer Aided Design of Integrated Circuits and Systems, 19 (2000), pp. 425–436.

[26] W. SHI, J. LIU, N. KAKANI, AND T. YU, *A fast hierarchical algorithm for 3-d capacitance extraction*, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 21 (2002), pp. 330–336.

[27] J. P. SINGH, C. HOLT, T. TOTSUKA, A. GUPTA, AND J. L. HENNESSY, *Load balancing and data locality in hierarchical n-body methods*, Journal of Parallel and Distributed Computing, 27 (1995), pp. 118–141.

[28] K. SRINIVASAN, H. MAHAWAR, AND V. SARIN, *A multipole based treecode using spherical harmonics for potentials of the form* $r^{-\lambda}$, in Proceedings of the 5^{th} International Conference on Computational Science, vol. 3514 of Lecture Notes in Computer Science, Springer-Verlag, Atlanta, Georgia, May 2005, pp. 107–114.

[29] S. H. TENG, *Provably good partitioning and load balancing algorithms for parallel adaptive n-body simulation*, SIAM Journal of Scientific Computing, 19 (1998), pp. 635–656.

[30] Y. YAUN AND P. BANERJEE, *A parallel implementation of a fast multipole based 3-d capacitance extraction program on distributed memory multicomputers*, Journal of Parallel and Distributed Computing, 61 (2001), pp. 1751–1774.

VITA

Hemant Mahawar was born in Cuttack, India in 1978. He completed his Bachelor of Technology in electrical engineering from Indian Institute of Technology (IIT), Bombay, India in May 2000. He joined the graduate program in the Department of Computer Science at Texas A&M University, College Station in Fall 2000 and earned his Master of Computer Science degree in December 2002. He continued his studies in Computer Science at Texas A&M University, obtaining his Ph.D. in May 2006.

During his stay at Texas A&M University, Hemant Mahawar worked as a Graduate Research Assistant for Dr. Vivek Sarin and Graduate Teaching Assistant for Dr. Walter Daugherity. His research interests are in scientific computing, parallel applications and numerical algorithms. He also published several articles in international journals and conferences such as: Parallel Computing, SIAM Journal of Scientific Computing, Design Automation Conference, and International Parallel and Distributed Processing Symposium among others. He served as the President of Computer Science Graduate Student Association in 2003 and Vice-president in 2002. His permanent address is: B-8, Apollo Apartments, Sector-3, Vidhyadhar Nagar, Jaipur, Rajasthan, India, 302023.

The typist for this thesis was Hemant Mahawar.