Fast Inductance Extraction of Large VLSI Circuits*

Hemant Mahawar
mahawarh@cs.tamu.edu

Vivek Sarin
sarin@cs.tamu.edu

Weiping Shi
wshi@cs.tamu.edu

Texas A&M University
College Station, TX 77843

Abstract

Accurate estimation of signal delay is critical to the design and verification of VLSI circuits. At very high frequencies, signal delay in circuits with small feature sizes is dominated by the parasitic inductance and capacitance of the interconnect. Inductance extraction involves the solution of large, dense, complex linear systems of equations by preconditioned iterative methods. Fast inductance extraction requires effective, parallelizable preconditioners for the system matrix which is available only implicitly via approximate hierarchical matrix-vector products. This paper presents a novel algorithm to solve these linear systems by restricting current to a discrete solenoidal subspace and solving the reduced system via an iterative method. A preconditioner based on the Green's function is suggested for accelerating the convergence of the iterative method. The paper outlines a parallelization scheme for matrix-vector products with the system matrix as well as the preconditioner. Experimental results are presented to show the advantages of the preconditioning scheme over existing approaches. The experiments also illustrate the parallel efficiency achieved on the SGI Origin2000 multiprocessor.

1. Introduction

With increasing clock speed and decreasing feature size, the delays in VLSI circuits are playing a crucial role. For the next generation micro-processors with more than a billion transistors and hundreds of millions of segment interconnect, signal delays will be dominated by the parasitic resistance (R), capacitance (C) and inductance (L) of the interconnects. Moreover, as the operation frequency reaches the tera-hertz range, inductance will play an important role in timing verification.

*This work has been supported in part by NSF under the grants NSF-CCR 9984400, NSF-CCR 9972533, and NSF-CCR 0113668. Computational resources were provided by NCSA, University of Illinois at Urbana-Champaign.

While there are a number of algorithms and software packages available for estimating the parasitic R and C, there is only one software package called *FASTHENRY* [5] for inductance extraction. Since most of these software packages are available only for uniprocessor workstations, the size of problems that can be solved is severely limited. There is significant interest in developing fast, parallel algorithms for inductance extraction of large VLSI circuits.

The general technique for inductance extraction discretizes the conductors with a mesh in which edges represent current carrying filaments. Potential is defined at each node and the difference between potential at adjacent nodes represents the potential drop across the corresponding filament. The potential drop across a filament is due to its own impedance as well as pairwise mutual inductance from all other filaments. The linear systems representing this relation are large, complex and dense.

These systems are solved by iterative methods such as Generalized Minimum Residual (GMRES) method [10]. At each iteration one needs to compute the product of a dense matrix with a vector. These matrix-vector products are often computed by fast approximate hierarchical methods such as Fast Multipole Method (FMM) and its variants [2, 8, 9, 11] and the singular value decomposition method [6]. The success of the approach depends on the rate of convergence of the iterative method which can be accelerated by preconditioning the system.

In this paper, we propose a novel approach to solve the linear systems arising in inductance extraction. Section 2 describes the discretization of the integral equation formulation for magneto-quasi-static analysis of a multi-conductor system. In section 3, we propose a novel scheme called the solenoidal basis method, to solve the linear system and describe an effective preconditioning scheme. Section 4 outlines the parallel implementation of the proposed algorithm and section 5 presents experimental results.

Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS'02)
1530-2075/02 $17.00 © 2002 IEEE

2. Background and Problem Definition

The interconnect of VLSI circuits is modeled as a mesh which is subdivided into smaller filaments for more accurate representation. With the complex arrangement of components in modern chips, this representation essentially maps to two dimensional structure with several layers of interconnect stacked on top of each other.

The impedance of an s-conductor geometry can be summarized by an $s \times s$ impedance matrix $\tilde{\mathbf{Z}}$. The l^{th} column of the impedance matrix is determined as follows: a *unit* current is sent through conductor *l*, and zero current is specified in other conductors. The numerical value of the potential difference between the two ends of conductor *k* gives values of the matrix element $\tilde{\mathbf{Z}}_{\mathbf{kl}}$. The above procedure is repeated *s* times to compute all columns of $\tilde{\mathbf{Z}}$.

The current density **J** at a point r is related to the potential $\mathbf{\Phi}$ by the following integral equation [5]

$$\mathbf{J}(\mathbf{r}) + j\omega \int_V \frac{\mu}{4\pi} \frac{\mathbf{J}(\mathbf{r}')}{\|\mathbf{r} - \mathbf{r}'\|} dV' = -\nabla \mathbf{\Phi}(\mathbf{r}), \qquad (1)$$

where μ is the magnetic permeability, $\mathbf{r}$ is a three-dimensional position vector, ω is the frequency, $\|\mathbf{r} - \mathbf{r}'\|$ is the Euclidean distance between $\mathbf{r}$ and $\mathbf{r}'$, and $j = \sqrt{-1}$. The volume of conductors is denoted by V and incremental volume with respect to $\mathbf{r}'$ is denoted by dV'.

A numerical solution of (1) can be obtained by discretizing the conductors into n filaments $V_1, V_2, ..., V_n$. Assuming current flows along the length of the filament and current density is constant within each filament, a linear system of the following form is obtained:

$$[\mathbf{R} + j\omega\mathbf{M}]\,\mathbf{I} = \mathbf{V},$$

where $\mathbf{R}$ is an $n \times n$ diagonal matrix of self resistances with $\mathbf{R}_{kk} = \rho l_k / a_k$ in which ρ is the resistivity, l_k is the length of the kth filament, a_k is the cross-sectional area of the kth filament, $\mathbf{I}$ is the vector of branch currents, and $\mathbf{V}$ is the vector of branch potentials. Let $\mathbf{u_k}$ denote the unit vector along the kth filament V_k. The elements of the inductance matrix $\mathbf{M}$ are given by

$$\mathbf{M}_{kl} = \frac{\mu}{4\pi} \frac{1}{a_k a_l} \int_{r_k \epsilon V_k} \int_{r_l \epsilon V_l} \frac{\mathbf{u}_k . \mathbf{u}_l}{\|\mathbf{r}_k - \mathbf{r}_l\|} dV_k dV_l. \qquad (2)$$

The current satisfies Kirchoff's current law at each node according to the constraints

$$\mathbf{B}^T \mathbf{I} = \mathbf{I}_d,$$

where $\mathbf{B}^T$ is the $m \times n$ branch index matrix of filaments and nodes and $\mathbf{I}_d$ is the known branch current vector of length m with non-zero values corresponding to the source currents. Assuming $\mathbf{V}_d$ is the unknown node potential vector of length m, the following linear system must be solved

$$\begin{bmatrix} \mathbf{R} + j\omega\mathbf{M} & -\mathbf{B} \\ \mathbf{B}^T & 0 \end{bmatrix} \begin{bmatrix} \mathbf{I} \\ \mathbf{V}_d \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{I}_d \end{bmatrix}.$$

A straightforward approach to solve this system involves removal of $\mathbf{I}$ by a block-step of Gaussian elimination. The resulting system is defined in terms of the unknowns $\mathbf{V}_d$ only, and can be expressed as

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

When solving this system by an iterative method, each step involves the solution of $[\mathbf{R} + j\omega\mathbf{M}]\mathbf{z} = \mathbf{d}$ via an inner iterative scheme, resulting in expensive outer iterations. Furthermore, with this kind of a complicated system matrix, it is difficult to find efficient preconditioners for the outer solver.

3. An Iterative Algorithm for Computing Inductance

3.1. The Solenoidal Basis Approach

The solenoidal basis method is used to solve the problem

$$\begin{bmatrix} \mathbf{Z} & -\mathbf{B} \\ \mathbf{B}^T & 0 \end{bmatrix} \begin{bmatrix} \mathbf{I} \\ \mathbf{V}_d \end{bmatrix} = \begin{bmatrix} \mathbf{F} \\ \mathbf{0} \end{bmatrix},$$

where $\mathbf{Z} = \mathbf{R} + j\omega\mathbf{M}$ is the impedance matrix, $\mathbf{M}$ is an $n \times n$ dense, symmetric and positive definite matrix, $\mathbf{B}$ is an $n \times m$ sparse matrix, and $\mathbf{R}$ is a diagonal matrix. The filament current vector $\mathbf{I}$ has size n, the node potential vector $\mathbf{V}_d$ has size m, and the right hand side vector $\mathbf{F}$ has size n. The overall system is complex and symmetric, with the complex part restricted to $\mathbf{Z}$. The vector $\mathbf{F}$ is the residual of the particular solution $[\mathbf{I}_p^T, 0]^T$ that satisfies the constraints $\mathbf{B}^T \mathbf{I}_p = \mathbf{I}_d$, and is given by

$$\mathbf{F} = -\mathbf{Z}\mathbf{I}_p.$$

Such a particular solution, $\mathbf{I}_p$, can easily be found by number of techniques. The simplest way is to assign $unit$ current to filaments on a path from node with input source current to the node with output source current (see, e.g., Fig. 1). This approach gives a sparse vector $\mathbf{I_p}$.

Since a unit current flowing in a closed loop in the mesh satisfies Kirchoff's law, it also satisfies the constraints imposed by $\mathbf{B}^T$. The solenoid basis method uses these mesh currents to represent the unknown current $\mathbf{I}$. This scheme is similar to the mesh current approach proposed in [3, 4] with the exception of the treatment of source current. In addition, the preconditioning proposed in section 3.2 is more powerful than those suggested in [4].

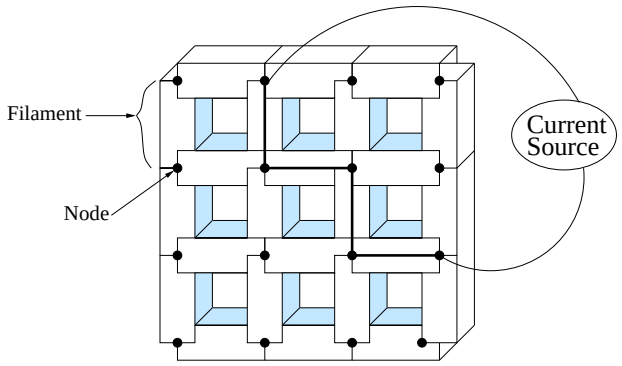

Figure 1. Discretization of a ground plane with a mesh of filaments. Current flowing through the filaments must satisfy Kirchoff's law at each node in the mesh. The bold line indicates a path for current that satisfies boundary conditions.

The null space of $\mathbf{B}^T$ represents a basis for current that obeys Kirchoff's law. Any full-rank matrix $\mathbf{P} \in R^{n\times(n-m)}$, that satisfy $\mathbf{B}^T\mathbf{P} = 0$ can be used to compute $\mathbf{I}$ via the matrix-vector product: $\mathbf{I} = \mathbf{P}\mathbf{x}$ for arbitrary $\mathbf{x} \in R^{(n-m)}$. A purely algebraic approach such as QR factorization of $\mathbf{B}$ cannot be used to compute $\mathbf{P}$ due to the prohibitive cost of computation and storage. We define a *unit* current flow in a closed loop as a *local* solenoidal function. Each such mesh current is represented as a vector and the set of these vectors forms the columns of $\mathbf{P}$. The resulting system is sparse due to localized structure of the solenoidal functions. Furthermore, operations such as computation and storage of $\mathbf{P}$ and matrix-vector products with $\mathbf{P}$, can be implemented efficiently in parallel.

Restricting the current vector to the subspace of $\mathbf{P}$ automatically satisfies the constraint

$$\mathbf{B^T P x} = \mathbf{B^T I} = 0.$$

Thus, we only need to solve

$$\mathbf{ZPx - BV_d = F}.$$

After eliminating the branch potential unknowns $\mathbf{V}_d$ by multiplying this equation with $\mathbf{P}^T$, we get

$$\mathbf{P^T Z P x = P^T F}. \tag{3}$$

Equating the real and imaginary components of (3), we get the following real-valued system

$$\begin{bmatrix} \mathbf{P^T R P} & -\omega\mathbf{P^T M P} \\ \omega\mathbf{P^T M P} & \mathbf{P^T R P} \end{bmatrix} \begin{bmatrix} \mathbf{a} \\ \mathbf{b} \end{bmatrix} = \begin{bmatrix} -\mathbf{P^T R I_p} \\ -\omega\mathbf{P^T M I_p} \end{bmatrix}, \tag{4}$$

where $\mathbf{x} = \mathbf{a} + \mathbf{jb}$.

This reduced system can be solved via a suitable iterative scheme such as the GMRES method. Once $\mathbf{x}$ is obtained, current is computed as $\mathbf{I} = \mathbf{P}\mathbf{x}$, and branch potential vector is recovered by solving the least squares problem

$$\mathbf{BV}_d \approx \mathbf{ZI - F}.$$

The potential difference between two nodes is computed by adding the potential drops across filaments on any path connecting the nodes. The right hand side vector $\mathbf{ZI - F}$ gives the potential drop across the filaments. Thus, it is not necessary to solve the above least squares problem to compute potential difference between any two nodes.

3.2. Preconditioning the Linear System

To accelerate the convergence of the iterative method, one must devise robust and effective preconditioning schemes for the linear system (4). Matrix-vector products with the system matrix $\mathbf{P^T Z P}$ are computed as a sequence of three products:

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

Of these, the most time consuming one is the product with the dense matrix $\mathbf{M}$ which is computed efficiently via approximate hierarchical methods such as those outlined in section 3.3. The task of designing effective preconditioners is made especially challenging due to the unavailability of $\mathbf{M}$. Popular techniques based on incomplete factorizations of the system matrix compute parts of the system matrix selectively, followed by incomplete factorizations. These schemes tend to be expensive and limited in their effectiveness [4].

Accurate preconditioning is critical to the success of the solenoidal basis method. A preconditioner is said to be optimal when the condition number of the preconditioned system, i.e., the ratio of the largest and the smallest singular values, is bounded by a constant. In such a case, the number of iterations of GMRES are fixed regardless of the frequency ω and mesh width h. The system matrix $\mathbf{P^T Z P}$ can be analyzed using the observation that the matrices $\mathbf{P}$ and $\mathbf{P^T}$ are equivalent to discrete curl operators. Since the system matrix is dominated by the inductance matrix $\mathbf{M}$ at high frequency operation, we use the following preconditioner for (4):

$$\mathbf{W} = \begin{bmatrix} \mathbf{0} & -\tilde{\mathbf{M}} \\ \tilde{\mathbf{M}} & \mathbf{0} \end{bmatrix}, \tag{5}$$

where $\tilde{\mathbf{M}}$ is an $(n-m)\times(n-m)$ matrix defined as follows:

$$\tilde{\mathbf{M}}_{\mathbf{ij}} = \frac{\omega\mu}{4\pi}\int_V \frac{1}{\|\mathbf{r_i} - \mathbf{r_j}\|}dV', \tag{6}$$

where $\mathbf{r_i}$ and $\mathbf{r_j}$ are the centers of loops i and j in the mesh. The preconditioning step consists of the matrix-vector product $\mathbf{z} = \mathbf{Wr}$ which can be computed using approximate hierarchical techniques identical to those used for $\mathbf{M}$.

There are several advantages of our preconditioning approach. The preconditioning step requires a matrix-vector product which is relatively inexpensive compared to incomplete factorization based preconditioners. The latter involve incomplete factorizations of partially computed $\mathbf{P^T ZP}$ and triangular solves which are expensive, especially on parallel platforms. In addition, experimental evidence suggests that the preconditioner is robust and very effective for the high frequency regime. Experiments in section 5 demonstrate the numerical and computational superiority of the preconditioner over existing techniques. This preconditioning scheme is being extended to low frequency regime as well. Details of the preconditioning approach are presented in a forthcoming paper [7].

3.3. Approximate Hierarchical Methods for Matrix-Vector Products

Accurate matrix-vector products with the dense matrices $\mathbf{M}$ and $\tilde{\mathbf{M}}$ require $O(n^2)$ operations where n is the size of these matrices. There are a number of techniques that can be used to exploit the rapid decay of the kernel with distance in (2) and (6) to compute approximate matrix-vector products in $O(n \log n)$ or $O(n)$ operations. These include hierarchical techniques such as Barnes-Hut [1] and Fast Multipole Method (FMM) [2] which use a truncated series approximation of filament currents within a localized region to estimate impact on well-separated sets of filaments. The method of Barnes and Hut relies only on filament-cluster interactions to achieve an $O(n \log n)$ computational bound for uniform filament distributions. The Fast Multipole Method uses both filament-cluster and filament-filament interactions to achieve an $O(n)$ bound for uniform distributions. In each case, reduction in computational complexity is associated with decrease in accuracy of the matrix-vector product.

We use a variant of the Barnes-Hut algorithm developed specifically for the inductance extraction problem. A hierarchical quadtree is used to partition the filaments in the mesh. Internal nodes of the tree represent hypothetical filaments carrying current equal to the sum of currents in that subtree. The coordinates of these hypothetical filaments are computed as a weighted sum of coordinates of the subtree filaments in which the weights correspond to the magnitude of current flowing in those filaments. This strategy exploits the fact that the combined inductance effect of a cluster of filaments can be approximated by the inductance effect of a single filament placed at the "center of current" of that group. This approximation is used to compute inductance on a filament which is well-separated from the cluster. This is analogous to the center of mass concept in Barnes-Hut algorithm.

The leaf nodes of the quadtree are filaments of the mesh. Inductance on a leaf node is computed by traversing the tree in a top down manner. A subtree is traversed only if the corresponding box is not well-separated from the leaf node. Well-separatedness is established by using a distance metric to determines if the leaf node is sufficiently "far off" such that the error in the associated truncated expansion is below a threshold. The inductance due to filaments in a well-separated subtree is computed by direct interaction with the corresponding hypothetical filament. Remaining interactions consist of leaf-leaf interactions which consist of mutual inductance computation between pairs of filaments. Care should be taken to ensure that the algorithm doesn't check the well-separatedness criteria for the box containing the leaf node itself.

4. Parallel Formulation

The main components of the algorithm are matrix-vector products with $\mathbf{P}$ and $\mathbf{P^T}$, vector additions, inner-products and hierarchical approximate matrix-vector products with the dense matrices $\mathbf{M}$ and $\tilde{\mathbf{M}}$. Parallelization of these operations require that the underlying mesh to be distributed across processors. Since the computation in each component is a function of the submesh size on each processor, a load balancing scheme should create almost equal sized partitions. On a p processor machine, the mesh is divided into p partitions and the jth partition is given to processor j. The unknowns for filament currents, loop currents, and potential defined on the edges, cells, and nodes, respectively, within a partition j reside on processor j that owns the partition.

Parallel matrix-vector products with $\mathbf{P}$ and $\mathbf{P^T}$ require inter-processor communication among processors with neighboring submeshes. The choice of local solenoidal functions leads to a sparse P with non-zero pattern similar to the adjacency matrix of the mesh. The matrix-vector products Px and $P^T z$ can be computed concurrently in a manner similar to that of a product with the discrete Laplace matrix.

One can partition a mesh across processors in several ways. The amount of inter-processor communication needed in computing the matrix-vector products varies largely on the mesh partitioning scheme. For uniform grids, partitioning across any one dimension is the simplest approach. One can achieve this by a simple loop parallelization scheme by distributing the loop control index which corresponds to the dimension along which the grid is partitioned. The resulting communication is $O(n^{1-1/d})$ for a d-dimensional uniform grid with $n^{1/d}$ nodes along each dimension. For a 2D uniform grid with $n^{1/2}$ nodes along each

dimension, the inter-processor communication is $O(n^{1/2})$. An optimal partitioning for such a grid is obtained by partitioning the grid across all the dimensions equally.

Parallelization of the hierarchical algorithm for matrix-vector products with $\mathbf{M}$ and $\tilde{\mathbf{M}}$ is the most challenging step. Several approaches have been proposed for these hierarchical algorithms and their performance analyzed extensively. One can partition the quadtree across the processors by allocating the subtrees at an appropriate level of the quadtree to individual processors. The level can be chosen to realize tasks of specific granularity. Each processor initializes its subtree, followed by a cooperative distribution of tasks to initialize the remaining nodes. Once the quadtree data structure has been initialized, processor i is responsible for computing inductance for all the filaments in its subtree. In this manner, the parallel formulation exploits the concurrency available in independent tree traversals for each filament. The filaments can be sorted in a proximity-preserving order (a Peano-Hilbert ordering) and mutual inductance computation for appropriate filament clusters aggregated into a single task. The implementation can be optimized to improve data-locality across processors and eliminate false sharing.

5. Experimental Results

In this section, we show the effectiveness of our preconditioned solenoidal basis method for the ground-plane problem. This benchmark problem allows straightforward discretization by uniform two-dimensional meshes with varying mesh width h. This problem exhibits the dependence of the condition number of the system matrices on h via the growth in iterations of the unpreconditioned GMRES algorithm. Such convergence behavior provides an ideal problem to test the effectiveness of the preconditioning approach for varying levels of refinement as well as operating frequencies. The experiments reported here involve successive refining of the mesh from 32×32 filaments to 128×128 filaments in the frequency range of 1GHz–1THz.

A two dimensional ground plane of size 1cm $\times$ 1cm is discretized by a mesh of $k \times k$ filaments, where $k = 32$, 64 and 128. Each filament has length $L = 1/k$, breadth $B = L/3$, and thickness $T = 2^{-13}$ (all lengths are measured in cm). Table 1 shows the number of iterations needed by the preconditioned GMRES to solve the linear system (4). A tolerance of 10^{-4} was specified on the relative residual norm. The table shows that the rate of convergence is independent of the frequency ω and the mesh width h indicating that the preconditioner is optimal for the benchmark problem .

Mesh Size	Filament Length	Frequency (in GHz)			
		1	10	100	1000
33×33	2^{-5}	18	14	13	13
65×65	2^{-6}	27	16	16	16
129×129	2^{-7}	46	19	18	17

Table 1. Iterations for convergence of preconditioned GMRES.

5.1. Comparison with *FASTHENRY*

The benchmark problem was used to compare the performance of solenoidal basis method with that of *FASTHENRY*, the only public domain software for inductance extraction. *FASTHENRY* uses mesh currents to generate a reduced system which is then solved by the preconditioned GMRES method. Fast Multipole Method is used to compute matrix-vector products with dense matrices. Preconditioners are derived from incomplete factorizations of approximations to the reduced system. These approximations are computed by various techniques to sparsify the inductance matrix $\mathbf{M}$. The software allows preconditioners such as CUBE and SHELL, in which the off-diagonal non-zeros in each column of $\mathbf{M}$ are restricted to those resulting from mutual inductance between filaments in the same box at a specific level in the quad tree and mutual inductance between filaments within a specific radius, respectively. The DIAG preconditioner corresponds to the case when all mutual inductances are ignored.

For the solenoidal basis method, the Barnes-Hut variant described in section 3.3 was used to compute products with the system matrix $\mathbf{M}$ and preconditioner $\tilde{\mathbf{M}}$ 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.

Mesh	FASTHENRY		Solenoidal Method
	DIAG	CUBE	
33×33	20	17	12
65×65	25	22	14
129×129	31	28	17

Table 2. Comparison of the number of iterations needed for convergence of the preconditioned GMRES.

Table 2 shows the number of iterations needed by preconditioned GMRES to solve the linear system. The fre-

Mesh	*FASTHENRY-DIAG*			*FASTHENRY-CUBE*			Solenoidal Method	
	Prec	GMRES	Total	Prec	GMRES	Total	GMRES	Total
33×33	1.87	4.3	6.17	3.65	3.91	7.56	6.41	6.62
65×65	54.85	26.55	81.4	95.04	26.72	121.76	33.05	34.01
129×129	1337.38	146.31	1483.69	3953.85	141.55	4094.40	165.92	170.17

Table 3. Comparison of the execution time (in seconds).

Processors	**GMRES**		**Matvec**		**Preconditioning**		**Total**	
	Time	Speedup	Time	Speedup	Time	Speed up	Time	Speedup
32	0.1147	30.18	3.085	12.23	1.016	7.31	4.749	9.73
16	0.2885	12.00	4.331	8.72	1.150	6.46	6.192	7.46
8	0.3941	8.79	6.422	5.88	1.682	4.41	8.666	5.33
4	0.8746	3.96	10.980	3.44	2.467	3.01	14.136	3.27
2	1.6940	2.04	19.583	1.93	3.838	1.93	24.229	1.91

Table 4. Execution time and speed improvement for a problem with mesh size 64 $\times$ 64 obtained on the SGI Origin2000.

Processors	**GMRES**		**Matvec**		**Preconditioning**		**Total**	
	Time	Speedup	Time	Speedup	Time	Speedup	Time	Speedup
32	1.00	22.69	14.577	16.40	5.056	8.74	21.80	13.41
16	1.91	11.88	25.686	9.31	6.504	6.79	36.97	7.91
8	2.63	8.63	40.218	5.95	10.051	4.39	53.74	5.44
4	5.35	4.24	70.636	3.38	15.34	2.88	90.04	3.25
2	10.02	2.26	123.84	1.93	22.56	1.95	151.39	1.93

Table 5. Execution time and speed improvement for a problem with mesh size 128 $\times$ 128 obtained on the SGI Origin2000.

quency of operation was 10GHz. A tolerance of 10^{-4} was specified on the relative residual norm of both the methods. *FASTHENRY* was allowed to use default values for all other parameters. A growth in the number of iterations with mesh size indicates a sub-optimal preconditioning scheme which contributes an additional factor towards the increase in cost of solving these systems as the mesh size is increased. In contrast, the solenoidal basis method is able to solve the systems in almost fixed number of iterations.

Table 3 shows the time taken to solve the same linear systems. The table shows a breakup of time into the main steps: construction of the preconditioner (Prec), solution of the linear system via preconditioned GMRES (GMRES), and the total time. In the case of solenoidal method, no additional computation is required to construct the preconditioner. All parameters are identical to the experiments in Table 2. Although the solenoidal method is slightly more expensive per GMRES iteration, the fixed number of iterations allows it to be competitive with *FASTHENRY*. The modest performance of the preconditioners in *FASTHENRY* come with a significant cost of computing the preconditioners themselves. As a result, the overall time spent required by the solenoidal method is considerably less than *FASTHENRY*. This comparative advantage will grow significantly with larger mesh sizes.

5.2. Parallel Performance

The results on parallel performance of the solenoidal method are presented in Table 4 and 5. These experiments were conducted on a 32 processor SGI Origin2000 at NCSA, University of Illinois. The code was parallelized with OpenMP directives. Execution time and speedup in the main steps of the solenoidal method are reported. The speedup is computed as the ratio of the runtime of the threaded version with multiple kernel threads to that of the single thread version. The column for GMRES reports time and speedup for all operations in GMRES routine other than

the matrix-vector product and preconditioning.

Considering the diverse set of computations involved in the GMRES algorithm including vector operations and hierarchical matrix-vector products, the parallel performance of the code is very good on up to 8 processors. On more processors, the performance degrades due the collective effect of increased communication in the hierarchical matvec routine, multiple inner products, and the preconditioning routine. As expected, the parallel performance is marginally better for the larger problem in Table 5 due to higher computation to communication ratio per processor. Higher efficiency is expected on larger number of processors for realistic problems with billions of filaments.

Performance can be further improved by using the Fast Multipole Method instead of the Barnes-Hut variant in which cache-friendly computation can be increased by selecting a higher multipole degree parameter. This is accompanied by relatively smaller increase in communication which leads to much better parallel performance. In addition, this has the desirable effect of improving accuracy as well.

6. Conclusions

This paper presents a solenoidal basis method for inductance extraction of VLSI circuits. The proposed approach solves a linear system of equations to compute the mutual inductance effect on a filament mesh using eddy currents defined on the mesh loops. The resulting reduced system is solved iteratively by the preconditioned GMRES method. The preconditioner suggested for the system matrix exhibits convergence in a fixed number of iterations irrespective of the mesh refinement and operational frequency. Experimental results indicate that the serial performance of the algorithm is superior to existing methods. Furthermore, the parallel implementation demonstrates good parallel efficiency on modest number of processors. Significantly higher performance is expected on larger, realistic problems.

References

[1] J. Barnes and P. Hut. A hierarchical O(n log n) force calculation algorithm. *Nature*, Vol. 324(4):446–449, December 1986.

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

[3] M. Kamon. *Fast Parasitic Extraction and Simulation of Three-dimensional Interconnect via Quasistatic Analysis*. PhD thesis, EE & CS Dept., MIT, 1998.

[4] M. Kamon and J. R. Philips. Preconditioning techniques for constrained vector potential integral equations, with application to 3-D magnetoquasistatic analysis of electronic packages. In *Proceedings of the Colorado Conference on Iterative Methods*, April 1994.

[5] M. Kamon, M. J. Tsuk, and J. White. FASTHENRY: A multipole-accelerated 3D inductance extraction program. *IEEE Transaction on Microwave Theory and Techniques*, Vol. 42(9):1750–1758, September 1994.

[6] S. Kapur and D. Long. A fast integral equation solver for efficient 3-D extraction. In *Proc. IEEE International Conference on Computer Aided Design*, pages 448–455, November 1997.

[7] H. Mahawar and V. Sarin. A class of preconditioned iterative methods for fast inductance extraction. Tech. Report, Dept. of Computer Science, Texas A&M University (in preparation).

[8] K. Nabors and J. White. Fastcap: A multipole accelerated 3-D capacitance extraction program. *IEEE Transcations on Computer Aided Design*, Vol. 10(11):1447–1459, November 1991.

[9] V. Rokhlin. Rapid solution of integral equations of classical potential theory. *Journal of Computational Physics*, Vol. 60(2):187–207, September 1985.

[10] Y. Saad. *Iterative methods for sparse linear systems*. PWS publishing company, Boston, massachusetts, 1996.

[11] W. Shi, J. Liu, N. Kakani, and T. Yu. A fast hierarchical algorithm for 3-D capacitance extraction. In *Design Automation Conference*, pages 212 – 217, June 1998.