Sparse Transformations and Preconditioners for Hierarchical 3-D Capacitance Extraction with Multiple Dielectrics *

Shu Yan
Texas A&M University
College Station, TX 77843
shu@ee.tamu.edu

Vivek Sarin
Texas A&M University
College Station, TX 77843
sarin@cs.tamu.edu

Weiping Shi
Texas A&M University
College Station, TX 77843
wshi@ee.tamu.edu

ABSTRACT

Capacitance extraction is an important problem that has been extensively studied. This paper presents a significant improvement for the fast multipole accelerated boundary element method. We first introduce an algebraic transformation to convert the $n \times n$ dense capacitance coefficient matrix into a sparse matrix with $O(n)$ nonzero entries. We then use incomplete Cholesky factorization or incomplete LU factorization to produce an effective preconditioner for the sparse linear system. Simulation results show that our algorithm drastically reduces the number of iterations needed to solve the linear system associated with the boundary element method. For the $k \times k$ bus crossing benchmark, our algorithm uses 3-4 iterations, compared to 10-20 iterations used by the previous algorithms such as FastCap [1] and HiCap [2]. As a result, our algorithm is 2-20 times faster than those algorithms. Our algorithm is also superior to the multi-scale method [3] because our preconditioner reduces the number of iterations further and applies to multiple dielectrics.

Categories and Subject Descriptors: B.7.2 [Integrated Circuits]: Design Aids — *simulation, verification*

General Terms: Algorithms.

Keywords: Capacitance extraction, interconnect, iterative methods, preconditioning.

1. INTRODUCTION

The Boundary Element Method (BEM) is the basis for many capacitance extraction algorithms. FastCap [1], HiCap [2], multi-scale [3], and other algorithms [4] are based on BEM and accelerated with Fast Multiple Method (FMM). The pre-corrected FFT algorithm [5] and the singular value decomposition algorithm [6] are based on BEM but not FMM. The linear system arising from BEM is dense and often solved by iterative methods. However for these dense matrices, cost-effective preconditioners are difficult to construct.

In this paper, we propose an algebraic linear transformation to convert the dense linear system from HiCap [2], into an equivalent sparse system. The transformed linear system is solved by CG or GMRES iterative methods. The sparse structure is exploited to construct preconditioners based on incomplete LU or incomplete Cholesky factorization. The rate of convergence of the iterative methods increases dramatically through the use of these preconditioners. Experimental results show that our algorithm is significantly faster than all previous methods, including FastCap [1], SVD [6], pre-corrected FFT [5], and multi-scale [3] on several standard benchmarks.

In Section 2, we review the integral equation approach for capacitance extraction for uniform and multiple dielectrics, and HiCap algorithm [2]. In Section 3, we introduce the new algorithm. We present experimental results in Section 4 and conclusions in Section 5.

2. PRELIMINARY

The capacitance of an m-conductor geometry is summarized by an $m \times m$ capacitance matrix $\mathbf{C}$. To determine the j-th column of the capacitance matrix, we compute the surface charges on each conductor produced by raising conductor j to unit potential while grounding the other conductors. Then C_{ij} is numerically equal to the charge on conductor i. This procedure is repeated m times to compute all columns of $\mathbf{C}$.

2.1 Uniform Dielectric

For conductors in the uniform media, each of the m potential problems can be solved using an equivalent free-space formulation where the conductor-dielectric interface is replaced by a charge layer of density σ. The charge layer satisfies the integral equation

$$\psi(x) = \int_{surfaces} \sigma(x') \frac{1}{4\pi\epsilon_0 \|x - x'\|} d\alpha', \qquad x \in surfaces, \tag{1}$$

where $\psi(x)$ is the known conductor surface potential, $d\alpha'$ is the incremental conductor surface area, $x' \in d\alpha'$, and $\|x - x'\|$ is the Euclidean distance between x and x'.

*This research was supported by the NSF grants CCR-0098329, CCR-0113668, EIA-0223785, and ATP grant 512-0266-2001.

DAC 2004, June 7–11, 2004, San Diego, California, USA.

A Galerkin scheme is often used to numerically solve (1) for σ. In this approach, the conductor surfaces are divided into n small panels, $A_1, \ldots, A_n$, and a dense linear system is built:

$$\mathbf{P_{cc}q_c} = \mathbf{v_c}, \tag{2}$$

where $\mathbf{q_c} \in \Re^n$ is the unknown vector of conductor panel charges, $\mathbf{v_c} \in \Re^n$ is the vector of known conductor panel potentials, and $\mathbf{P_{cc}} \in \Re^{n \times n}$ is the potential coefficient matrix. Each entry of $\mathbf{P_{cc}}$ is defined as

$$p_{ij} = \frac{1}{a_i}\frac{1}{a_j}\int_{A_i}\int_{A_j} \frac{1}{4\pi\epsilon_0\|x_i - x_j\|}\, d\alpha_j d\alpha_i \tag{3}$$

for panels A_i and A_j, where a_i and a_j are panel areas.

2.2 Multiple Dielectrics

For solving the multiple dielectrics problems, we apply the equivalent charge approach [7, 8] which uses the free-space Green's function in conjunction with total charge on the conductor surfaces and polarization charge on the dielectric-dielectric interfaces. Thus, the potential produced is given by

$$\begin{aligned}\psi(x) &= \int_{S_c} \sigma_c(x')\frac{1}{4\pi\epsilon_0\|x - x'\|}d\alpha' \\ &+ \int_{S_d} \sigma_d(x')\frac{1}{4\pi\epsilon_0\|x - x'\|}d\alpha', \end{aligned}\tag{4}$$

where σ_c and σ_d are the charge densities on the conductor surfaces S_c and the dielectric-dielectric interfaces S_d, respectively. The interface condition

$$\epsilon_a \frac{\partial\psi_a(x)}{\partial n_a} - \epsilon_b \frac{\partial\psi_b(x)}{\partial n_a} = 0 \tag{5}$$

should be satisfied at any point x on the dielectric-dielectric interface. Here, ϵ_a and ϵ_b are the permittivities of the two adjacent regions a and b, n_a is the normal to the dielectric-dielectric interface at x pointing to dielectric a, and ψ_a and ψ_b are the potentials at x in the dielectrics a and b, respectively. Using the same numerical approach as that for the uniform case, we transform (4) and (5) into the linear system

$$\begin{bmatrix} \mathbf{P_{cc}} & \mathbf{P_{cd}} \\ \mathbf{E_{dc}} & \mathbf{E_{dd}} \end{bmatrix} \begin{bmatrix} \mathbf{q_c} \\ \mathbf{q_d} \end{bmatrix} = \begin{bmatrix} \mathbf{v_c} \\ 0 \end{bmatrix}, \tag{6}$$

where $\mathbf{q_c}$ and $\mathbf{q_d}$ are the charge vectors of the conductor panels and dielectric-dielectric interface panels, respectively, and $\mathbf{v_c}$ is the potential vector of conductor panels. The entries of $\mathbf{P_{cc}}$ and $\mathbf{P_{cd}}$ are defined in (3). The entries of $\mathbf{E_{dc}}$ and off-diagonal entries of $\mathbf{E_{dd}}$ are defined as

$$e_{ij} = (\epsilon_a - \epsilon_b)\frac{\partial}{\partial n_a}\frac{1}{a_i}\frac{1}{a_j}\int_{A_i}\int_{A_j} \frac{1}{4\pi\epsilon_0\|x_i - x_j\|}\, d\alpha_j d\alpha_i,$$

and the diagonal entries of $\mathbf{E_{dd}}$ are defined as

$$e_{ii} = (\epsilon_a + \epsilon_b)\frac{1}{2a_i\epsilon_0}.$$

2.3 HiCap Algorithm

For the convenience of discussion in later sections, HiCap algorithm [2] is briefly introduced here. The algorithm constructs a hierarchical data structure for the potential coefficient matrix by adaptively subdividing the conductor surfaces into panels to ensure that the coefficients are approximated to within a user-supplied tolerance.

Fig. 1 shows the hierarchical data structure of a potential coefficient matrix. The panels are stored as nodes in the tree, and the coefficients are stored as links between the nodes. Each tree represents a single conductor surface whose surface panel is stored as the root node of the tree. Each non-leaf node represents a panel that is further subdivided into two child panels, which are stored as child nodes in the tree. Each leaf node represents a panel that is not further subdivided. The union of all the leaf nodes completely cover the surfaces of the conductors. The coefficient matrix determined by the links of Fig. 1 is a block matrix with $O(n)$ block entries [2], where n is the number of leaf panels.

The potential on each leaf panel is evaluated in three steps, which is the same as to compute the matrix-vector product $\mathbf{Pq}$. The first step, `AddCharge`, computes the charge of every panel in the tree through a depth-first traversal of the tree that propagates the charge upward. The second step, `CollectPotential`, computes the potential at each panel A_i due to its interacting panels by adding up the product of potential coefficient p_{ij} with charge at panel A_j, for each A_j that has an interaction with A_i. The third step, `DistributePotential`, distributes the potential from the non-leaf nodes to the leaf nodes through another depth first traversal of the tree that propagates potential down to the leaf nodes.

3. NEW ALGORITHM

The linear system arising in the capacitance extraction for both uniform dielectric (2) and multiple dielectrics (6) has the following form

$$\mathbf{Pq} = \mathbf{v}. \tag{7}$$

In this section, we introduce the new algorithm that transforms the dense system (7) to a sparse system, which is then solved by a preconditioned iterative method. Our discussion is based on HiCap [2]. The algorithm is outlined below.

New Algorithm

1. Construct the factorization $\mathbf{P} = \mathbf{J^T H J}$.
2. Transform the dense system (7) to equivalent sparse system $\tilde{\mathbf{P}}\tilde{\mathbf{q}} = \tilde{\mathbf{v}}$.
3. Compute a preconditioner for $\tilde{\mathbf{P}}$.
4. Solve $\tilde{\mathbf{P}}\tilde{\mathbf{q}} = \tilde{\mathbf{v}}$ via preconditioned CG or GMRES.
5. Compute capacitance.

3.1 Constructing Factorization of P (Step 1)

Consider the coefficient matrix implied by the hierarchical data structure of HiCap [2] in Fig. 1. Let n be the number of leaf panels and N be the total number of leaf and non-leaf panels. Let $\mathbf{q}, \mathbf{v} \in \Re^n$ denote the charge and potential vectors of the leaf panels, respectively. Let $\mathbf{H} \in \Re^{N \times N}$ be the

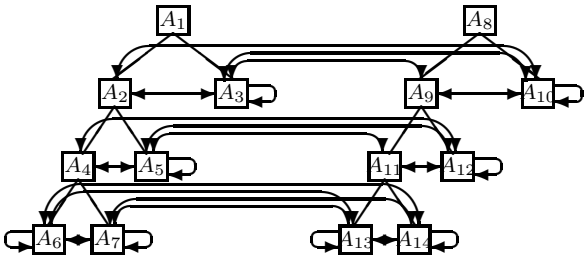


Figure 1: Hierarchical data structure and potential coefficients.

matrix of links among panels in the hierarchical data structure, including both leaf and non-leaf panels. For any two panels with no links in between, the corresponding entries in $\mathbf{H}$ are zero. Clearly, $\mathbf{H}$ is a sparse matrix with the number of nonzero entries equal to the number of block entries in $\mathbf{P}$. Let $\mathbf{J} \in \Re^{N \times n}$ represent the tree structure. Each row of $\mathbf{J}$ corresponds to a panel, either leaf or non-leaf, and each column corresponds to a leaf panel. Each entry $\mathbf{J}_{i,j}$ is 1 if panel i contains the leaf panel j, and is 0 otherwise. Let $\mathbf{q_N}, \mathbf{v_N} \in \Re^N$ be the charge and potential vectors of the N panels, respectively, where $\mathbf{v_N}$ is due to the corresponding interacting panels. The three steps for matrix-vector multiplication in Section 2.3 can be written as

- `AddCharge`: $\mathbf{q_N} = \mathbf{Jq}$,
- `CollectPotential`: $\mathbf{v_N} = \mathbf{Hq_N}$, and
- `DistributePotential`: $\mathbf{v} = \mathbf{J^T v_N}$.

The three steps can be combined to obtain the factorization

$$\mathbf{P} = \mathbf{J^T H J}.$$

3.2 Transforming Linear System (Step 2)

3.2.1 Overview

Given the matrix $\mathbf{J}$, we can construct an orthogonal transformation $\mathbf{F} \in \Re^{N \times N}$ (described later in this section), such that

$$\mathbf{FJ} = \begin{bmatrix} \mathbf{W} \\ \mathbf{0} \end{bmatrix},$$

where $\mathbf{W} \in \Re^{n \times n}$. Thus,

$$\begin{aligned} \mathbf{P} &= \mathbf{J^T H J} \\ &= (\mathbf{FJ})^{\mathbf{T}} \left(\mathbf{FHF^T}\right) (\mathbf{FJ}) \\ &= \begin{bmatrix} \mathbf{W^T} & \mathbf{0} \end{bmatrix} \left(\mathbf{FHF^T}\right) \begin{bmatrix} \mathbf{W} \\ \mathbf{0} \end{bmatrix}, \end{aligned}$$

where $\mathbf{FHF^T}$ can be represented as

$$\mathbf{FHF^T} = \begin{bmatrix} \tilde{\mathbf{P}} & \times \\ \times & \times \end{bmatrix},$$

in which $\tilde{\mathbf{P}}$ is sparse (we show this property later). Since $\mathbf{P} = \mathbf{W^T \tilde{P} W}$, the linear system (7) can be transformed to the sparse system

$$\tilde{\mathbf{P}}\tilde{\mathbf{q}} = \tilde{\mathbf{v}}, \tag{8}$$

where $\tilde{\mathbf{q}} = \mathbf{Wq}$ and $\tilde{\mathbf{v}} = \mathbf{W^{-T} v}$.

3.2.2 Constructing F

We first introduce a basic transformation that is used to construct $\mathbf{F}$. Given

$$\hat{\mathbf{J}}_\mathbf{k} = \begin{bmatrix} 1 & 1 \\ c_k & 0 \\ 0 & c_k \end{bmatrix},$$

there exists an orthogonal matrix

$$\hat{\mathbf{F}}_\mathbf{k} = \begin{bmatrix} \sqrt{\frac{2}{{c_k}^2+2}} & \frac{c_k}{\sqrt{2({c_k}^2+2)}} & \frac{c_k}{\sqrt{2({c_k}^2+2)}} \\ \frac{c_k}{\sqrt{{c_k}^2+2}} & -\frac{1}{\sqrt{{c_k}^2+2}} & -\frac{1}{\sqrt{{c_k}^2+2}} \\ 0 & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix},$$

such that

$$\hat{\mathbf{F}}_\mathbf{k} \hat{\mathbf{J}}_\mathbf{k} = \begin{bmatrix} c_{k+1} & c_{k+1} \\ 0 & 0 \\ e_k & -e_k \end{bmatrix}, \tag{9}$$

where

$$c_k = \begin{cases} 1, & k = 1 \\ \sqrt{\frac{c_{k-1}^2+2}{2}}, & k > 1 \end{cases} \quad \textit{and} \quad e_k = \frac{c_k}{\sqrt{2}}.$$

For the convenience of discussion, we define `element tree` as the tree with one root and two children. The transformation for $\mathbf{J}$ is done by a depth-first traversal of the corresponding tree, propagating the transformation upward to the root. Fig. 2 illustrates the procedure.

Starting from height $k = 1$, as shown in Fig. 2(a), for each `element tree` rooted at height 1, i.e. tree (B, C, D) and (E, F, G), we can identify the corresponding $\hat{\mathbf{J}}_\mathbf{1}$ block in $\mathbf{J}$. We construct the transformation $\mathbf{F_1}$ which transforms all $\hat{\mathbf{J}}_\mathbf{1}$ blocks to $\hat{\mathbf{F}}_\mathbf{1}\hat{\mathbf{J}}_\mathbf{1}$ blocks and leaves the rest of $\mathbf{J}$ unchanged. Next, as illustrated in Fig. 2(b), we identify the `element tree` at height $k = 2$, i.e. tree (A, B, E), and corresponding $\hat{\mathbf{J}}_\mathbf{2}$ block in $\mathbf{F_1 J}$. Note that the rows for A, B and E have two instances of the $\hat{J}_2$ block in the columns C, F and D, G, respectively. We construct transformation $\mathbf{F_2}$ which transforms the $\hat{\mathbf{J}}_\mathbf{2}$ blocks to $\hat{\mathbf{F}}_\mathbf{2}\hat{\mathbf{J}}_\mathbf{2}$ blocks and leaves the rest of $\mathbf{F_1 J}$ unchanged. In this way, the transformation is propagated to root. Finally, as shown in Fig. 2(c), we permute the nonzero rows to the top part with the permutation matrix $\mathbf{E}$. The overall transformation is summarized as $\mathbf{F} = \mathbf{E F_2 F_1}$. It is easy to see that in $\mathbf{FJ}$, only rows corresponding to the root or the right child nodes are nonzero.

Generally, for a tree of height h, the transformation is

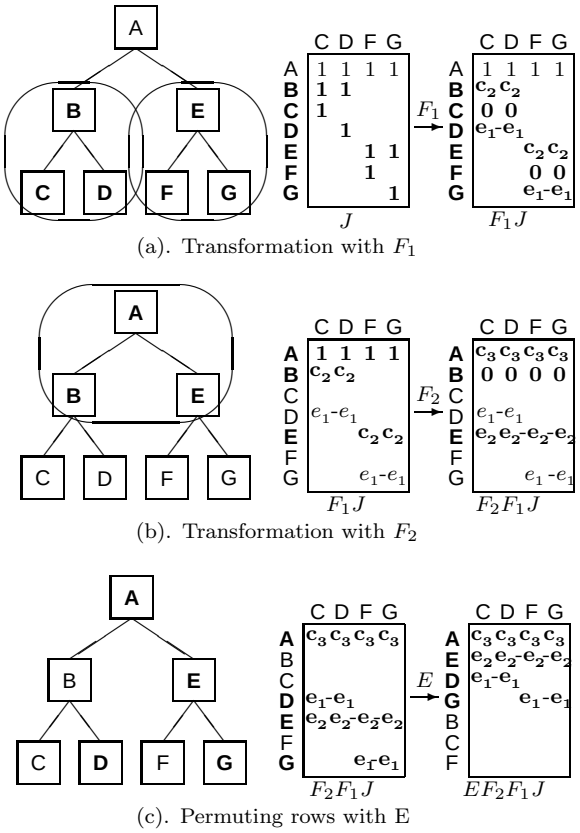
$$\mathbf{F} = \mathbf{E F_h F_{h-1}} \cdots \mathbf{F_2 F_1},$$

where $\mathbf{F_k}$ is constructed according to the element trees of height k. Since all $\mathbf{F_k}$s and $\mathbf{E}$ are orthogonal, $\mathbf{F}$ is orthogonal.

3.2.3 Computing FHF^T

The matrix $\mathbf{H}$ is transformed by applying the transformations $\mathbf{F_k}, k = 1, \ldots, h$ in order, as shown below

$$\mathbf{H_{k+1}} = \mathbf{F_k H_k F_k^T}, \qquad k = 1, \ldots, h,$$



(a). Transformation with F_1

(b). Transformation with F_2

(c). Permuting rows with E

Figure 2: The procedure of constructing the transformation F for a tree of height 2.

where $\mathbf{H_1} = \mathbf{H}$, and then by applying the permutation matrix

$$\mathbf{FHF^T} = \mathbf{EH_{h+1}E^T}.$$

With the hierarchical data structure, the transformation is done by a depth-first traversal of the tree, propagating the transformation upward, in a manner similar to the process of constructing the transformation matrix $\mathbf{F}$. The procedure is described below.

```
H2Pt(Panel Ai)
{
  if (Ai is leaf)
    return;
  if (Ai has children) {
    H2Pt(Ai.left);
    H2Pt(Ai.right);
  }
  ElementTreeTransform(Ai);
}
```

Here, the subroutine `ElementTreeTransform(Ai)` updates the three rows of $\mathbf{H}$, corresponding to `Ai`, `Ai.left` and `Ai.right`, by multiplying the rows by $\hat{\mathbf{F}}_\mathbf{k}$ from left side, where k is the height of `Ai` in the tree. The subroutine also updates the three columns of $\mathbf{H}$, corresponding to `Ai`, `Ai.left` and `Ai.right`, by multiplying the columns by $\hat{\mathbf{F}}_\mathbf{k}^\mathbf{T}$ from right side .

In $\mathbf{FHF^T}$, only the submatrix $\tilde{\mathbf{P}}$, which contains the transformed links among root nodes and right child nodes, is our concern. The matrix $\tilde{\mathbf{P}}$ is sparse because the number of nonzero entries in $\tilde{\mathbf{P}}$ is comparable to the number of block entries in $\mathbf{P}$, which is $O(n)$. This has been observed in the simulations as well (see, e.g., Fig. 3).

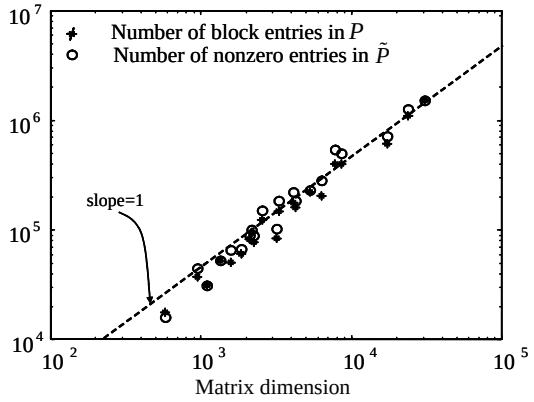


Figure 3: Number of nonzero entries in $\tilde{\mathbf{P}}$ and the number of block entries in $\mathbf{P}$ for different problem sizes.

3.2.4 *Computing* $\tilde{\mathbf{v}}$

The rows of the transformed matrix $\hat{\mathbf{F}}_\mathbf{k}\hat{\mathbf{J}}_\mathbf{k}$ are orthogonal for all k. It follows that the rows of $\mathbf{W}$ are orthogonal and $\mathbf{WW^T}$ is a diagonal matrix which can be easily inverted. This property is exploited when computing $\tilde{\mathbf{v}}$:

$$\tilde{\mathbf{v}} = \mathbf{W^{-T}v} = \left(\mathbf{WW^T}\right)^{-1}\mathbf{Wv}.$$

Moreover, the sum of entries in each row of $\mathbf{W}$ is zero except for the rows corresponding to the root nodes of the trees for the interface and conductor surface panels. As a result, $\tilde{\mathbf{v}}$ has nonzero entries of value ${c_{h+1}}^{-1}$ only at the root rows which correspond to the conductor surfaces with 1 volt, where h is the height of the root.

3.3 Solving Transformed System (Steps 3–4)

For the problem with uniform media, the sparse linear system (8) is symmetric. We use incomplete Cholesky factorization with no fill to compute the preconditioner. Preconditioned Conjugate Gradients method is used to solve the system. For the problem with multiple dielectrics, the sparse linear system is unsymmetric. The preconditioner is computed from incomplete LU factorization of the coefficient matrix. No fill is allowed during factorization. We use right preconditioned GMRES method to solve the system.

3.4 Computing Capacitance (Step 5)

We compute the capacitance directly by adding the entries of $\tilde{\mathbf{q}}$ at the root nodes of each conductor, scaled by ${c_{h+1}}^{-1}$. This is valid because the entries of $\tilde{\mathbf{q}} = \mathbf{Wq}$ corresponding to root nodes are the sum of all leaf panel charges of the trees, scaled by c_{h+1}.

3.5 Complexity Analysis

The complexity of constructing the factorization of P in Step 1 is $O(n)$ [2]. Transformation of the linear system in

Step 2 usually takes $O(n \log n)$ time. Since the number of nonzeros in $\tilde{P}$ is $O(n)$, incomplete factorization can be done in $O(n)$ operations. Each iteration requires a matrix-vector product with $\tilde{P}$ and two triangular solves with the factors of the preconditioner. Thus, Steps 3 and 4 take $O(n)$ time when the number of iterations is small. The computation of capacitance in Step 5 takes linear time. The overall complexity of this algorithm is $O(n \log n + mn)$, where m is the number of conductors.

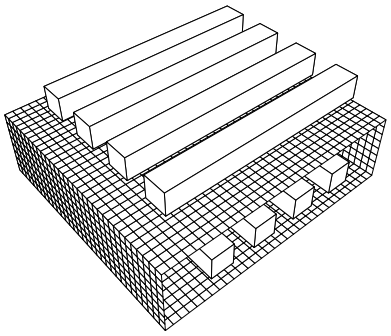


Figure 4: 4x4 bus with two layers of dielectrics (section view).

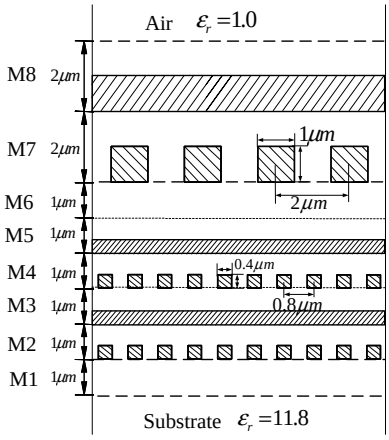


Figure 5: Example with 48 metal conductors and 8 dielectric layers. Metal wires are shaded. Relative permittivity of M1 is 3.9, M2 through M6 is 2.5, and M7 and M8 is 7.0. Layers M2 through M5 have 10 conductors each whereas layers M7 and M8 have 4 conductors each.

4. EXPERIMENTAL RESULT

Four algorithms are compared: FastCap [1] with expansion order 2, FastCap with expansion order 1, HiCap [2] and the new algorithm. In [2], HiCap algorithm can only solve problems in uniform media. To make the comparison complete, we extend HiCap to cover the multiple dielectrics.

The algorithms are executed on a Sun UltraSPARC Enterprise 4000. The uniform media examples are from [1]. In addition, we experiment with the $k \times k$ bus crossing in two dielectrics, as shown in Fig. 4. The media surrounding the upper layer conductors has permittivity of $3.9\epsilon_0$ and the lower layer conductors are in media with $7.5\epsilon_0$. The shaded box is the interface of the two media layers. Each bus is scaled to $1m \times 1m \times (2k+1)\,m$, where m is meter. The distance between buses in the same layer is $1m$ and the distance between the two bus layers is $2m$. We define the error of capacitance matrices $\mathbf{C}'$ as $\|\mathbf{C}-\mathbf{C}'\|_F/\|\mathbf{C}\|_F$, where $\|\bullet\|_F$ is the Frobenius norm.

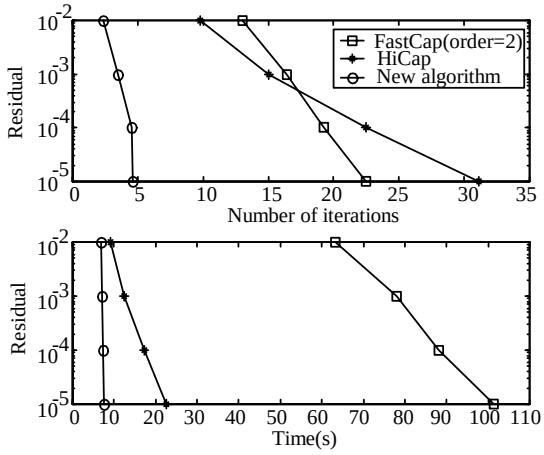


Figure 6: Comparison of convergence rate.

Table 1 compares the four algorithms. The new algorithm is the fastest and uses much less memory compared to FastCap. The error of the new algorithm is below 2.5%, which is satisfactory in practice. The new algorithm uses more memory than HiCap because of the extra storage for transformed sparse system. This can be improved with careful implementation.

We test on structures with more conductors and dielectric layers as shown in Fig. 5. Similar structures with 68 and 116 conductors are also tested. The experimental result of HiCap and new algorithm are reported in Table 2. We can not use FastCap to solve these examples because of prohibitive time and memory requirement.

Fig. 6 shows that the residual norm decreases rapidly for the new algorithm. In contrast, the decrease is slower for HiCap. As a result, the new algorithm requires much less time to solve the problem.

In addition to these experiments, we studied the performance of the new algorithm on the parallel-plate problem. It is well known that this problem yields ill-conditioned systems when the two plates are very close to each other. For a problem with plate size $20m \times 20m$ and distance between two plates $0.01m$, HiCap requires 67 iterations to converge. In contrast, the new algorithm needs only 5-6 iterations.

The multi-scale method [3] uses a similar idea to sparsify the dense matrix $\mathbf{P}$. However there are three differences with the proposed method. First, the multiscale method is based on high order FMM, whereas our method is based on HiCap. It was shown that the hierarchical approach in HiCap is more efficient and kernel independent [2]. Second, their method has been applied to uniform dielectric only, while our method is applied to multiple dielectrics. Finally, their preconditioner is block diagonal, while ours uses incomplete Cholesky or LU factorization. Based on the limited information in [3], we present a comparison of the number of iterations needed to reduce the residual norm below 10^{-9}

for $k \times k$ bus crossing benchmarks. Table 3 shows that the number of iterations required by the new algorithm is less than the multiscale method.

We also expect our algorithm to be faster than the precorrected FFT algorithm [5] and the SVD algorithm [6]. On standard benchmarks, these methods are only 2–3 times faster than FastCap, whereas our algorithm is much faster than FastCap.

Table 1: Comparison of four methods. Time is CPU seconds, iteration is average number of iterations for solving one conductor, memory is MB, convergence tolerance is 1%, and error is with respect to FastCap (order=2).

	FastCap (order=2)	FastCap (order=1)	HiCap	New algorithm
4x4 Bus with Uniform Dielectric				
Time	18.63	19	1.3	1.2
Iteration	8	14.9	9	3
Memory	25.7	16.7	1.6	2.4
Error	—	0.05%	0.97%	1.07%
Panel	2736		2244	
6x6 Bus with Uniform Dielectric				
Time	113.9	68.5	2.4	1.5
Iteration	14.4	14.5	11.9	3.2
Memory	62.5	40.3	1.9	2.9
Error	—	1.12%	2.19%	2.25%
Panel	5832		3168	
8x8 Bus with Uniform Dielectric				
Time	206	204	29.2	9.7
Iteration	12	21.9	13.8	3.5
Memory	112	67	7.2	12.1
Error	—	1.04%	1.4%	1.45%
Panel	10080		8576	
4x4 Bus with Two Layer Dielectrics				
Time	63	36	1.7	2.4
Iteration	13	14	9	3
Memory	68	39	3.0	4.3
Error	—	0.69%	1.09%	1.02%
Panel	3456		2120	
6x6 Bus with Two Layer Dielectrics				
Time	162	104	10.4	6.6
Iteration	17.1	17	11.3	3
Memory	92	61	6.3	10.3
Error	—	0.58%	1.00%	1.29%
Panel	5448		4120	
8x8 Bus with Two Layer Dielectrics				
Time	324	197	32.0	15.0
Iteration	18	18	12.8	3
Memory	133	86.9	11.5	18.9
Error	—	0.0%	1.40%	1.45%
Panel	7968		6784	

5. CONCLUSION

This paper proposes a new algorithm to transform the dense block linear system arising in capacitance extraction into a sparse system which is then solved by a preconditioned iterative method. The sparse structure allows construction of inexpensive but highly effective preconditioners based on incomplete factorization techniques.

Table 2: Comparison of HiCap and the new algorithm. Time is CPU seconds, iteration is average iteration number for solving one conductor, memory is MB, convergence tolerance is 1%. FastCap is not included because of the prohibitive time and memory requirements.

	48 conductors		68 conductors		116 conductors	
	HiCap	New alg.	HiCap	New alg.	HiCap	New alg.
Time	533	139	3011	650	12930	2875
Iteration	18.7	3.9	25.3	3.8	36.8	5.3
Memory	43	68	115	188	406	733
Panel	19840		42912		138552	

Table 3: Comparison of iteration numbers for multiscale method and new algorithm. Experiments are for $k \times k$ bus crossing conductors. Convergence tolerance is 10^{-9}.

	1×1	2×2	4×4	6×6	8×8
Multiscale	12	17	18	18	18
New	7	14	13	14	16

The experimental results show that the new algorithm beats HiCap in terms of number of iterations and running time. But the memory requirement is larger than HiCap. Compared with FastCap, the new algorithm is about 20 times faster and uses only about 10 % memory.

The application of the new algorithm is not confined to the extraction based on HiCap. It is applicable to other techniques based on multipole methods, where the linear system is represented by block matrix.

6. ACKNOWLEDGMENTS

The authors thank Sani Nassif for suggestion and examples that improved the presentation.

7. REFERENCES

[1] K. Nabors and J. White, "FastCap: A multipole accelerated 3-d capacitance extraction program," *IEEE Trans. on CAD*, pp. 1447–1459, 1991.

[2] W. Shi, J. Liu, N. Kakani, and T. Yu, "A fast hierarchical algorithm for 3-d capacitance extraction," *IEEE Trans. on CAD*, pp. 330–336, 2002.

[3] J. Tausch and J. White, "A multiscale method for fast capacitance extraction," *Proc. DAC*, pp. 537–542, 1999.

[4] M. Beattie and L. Pileggi, "Electromagnetic parasitic extraction via a multipole method with hierarchical refinement," *Proc. ICCAD*, pp. 437–444, 1999.

[5] J. R. Phillips and J. White, "A precorrected FFT method for capacitance extraction of complicated 3-d structures," *IEEE Trans. CAD*, pp. 1059–1072, 1997.

[6] S. Kapur and D. E. Long, "IES^3: A fast integral equation solver for efficient 3-dimensional extraction," *Proc. ICCAD*, pp. 448–455, 1997.

[7] T. K. Sarkar S. M. Rao and R. F. Harrington, "The electrostatic field of conducting bodies in multiple dielectric media," *IEEE Trans. on Microwave Theory Tech.*, pp. 1441–1448, 1984.

[8] K. Nabors and J. White, "Multipole-accelerated capacitance extraction algorithm for 3-d structures with multiple dielectrics," *IEEE Trans. on CAS*, pp. 946–954, 1992.