Algorithms in FastImp: A Fast and Wideband Impedance Extraction Program For Complicated 3-D Geometries *

Zhenhai Zhu, Ben Song and Jacob White

Department of Electrical Engineering and Computer Science
Massachusetts Institute of Technology, Cambridge, MA 02139
{zhzhu, bsong01, white}@mit.edu

ABSTRACT

In this paper we describe the algorithms used in FastImp, a program for accurate analysis of wide-band electromagnetic effects in very complicated geometries of conductors. The program is based on a recently developed surface integral formulation and a precorrected-FFT accelerated iterative method, but includes a new scaling and preconditioning technique as well as a generalized grid interpolation and projection strategy. Computational results are given on a variety of integrated circuit interconnect structures to demonstrate that FastImp is robust and can accurately analyze very complicated geometries of conductors.

Categories and Subject Descriptors

B.7.2 [**Integrating Circuits**]: Design Aids-placement and routing, simulation, verification

General Terms

Algorithms

Keywords

Impedance extraction, interconnect, fast integral equation solver, iterative methods, preconditioning

1. INTRODUCTION

Collocating sensitive analog circuits and rapidly switching digital logic on a single integrated circuit, as is typical in mixed signal designs, can create coupling problems that are very difficult to find and eliminate. The difficulty is that these coupling problems are often caused by simultaneous interactions between a large number of conductors. In order to help designers find these problems, there has been renewed emphasis on developing electromagnetic analysis tools capable of wide-band analysis of very complicated geometries of conductors.

In the area of electromagnetic analysis of complicated geometries of interconnect, most of the recently developed programs have been based on combining discretized integral formulations with accelerated iterative methods [1, 2, 3, 4, 5, 6, 7, 8]. Though these programs and techniques have been very effective, there is still no single program capable of solving full Maxwell's equations in general 3D structures with lossy conductors which is accurate from zero frequency to microwave frequencies.

In this paper we describe the algorithms used in FastImp, a program for accurate analysis of wide-band electromagnetic effects in very complicated geometries of conductors. The program is based on a recently developed surface integral formulation [9, 10, 11] and a precorrected-FFT accelerated iterative method [8], but includes a new scaling and preconditioning technique as well as a generalized grid interpolation and projection strategy for the precorrected-FFT method. The background for the surface formulation is described in the next section, and then the scaling and preconditioning approach is described in section 3. The generalized grid projection strategy is detailed in section 4. To demonstrate that FastImp is both fast and robust, a variety of examples are examined in section 5, and then conclusions are given in section 6.

*This work was supported by the Semiconductor Research Corporation, the Marco interconnect focus center, the DARPA neocad program, as well as the grants from the National Science Foundation and Intel Corporation.

DAC 2003, June 2–6, 2003, Anaheim, California, USA.

2. BACKGROUND

In this section, we summarize the surface formulation and review the discretization scheme used in [10]. The improvements we made in section 3 are closely tied with this discretization scheme.

2.1 Summary of a surface formulation

The formulation in [9] has as unknowns: $\vec{E}(\vec{r})$, the vector electric field at location $\vec{r}$; $\frac{\partial \vec{E}(\vec{r})}{\partial n}$, the normal derivative of the electric field; ϕ, the scalar potential; ρ, the charge density. It consists of the following equations

$$\frac{1}{2}\vec{E}(\vec{r}) = \int_{S_i} dS' \left(G_1(\vec{r},\vec{r'}) \frac{\partial \vec{E}(\vec{r'})}{\partial n(\vec{r'})} - \frac{\partial G_1(\vec{r},\vec{r'})}{\partial n(\vec{r'})} \vec{E}(\vec{r'}) \right), \quad \vec{r} \in S_i, \tag{1}$$

$$-\hat{t}(\vec{r}) \cdot \frac{1}{2}\vec{E}(\vec{r}) = \hat{t}(\vec{r}) \cdot \int_S dS' \left(G_0(\vec{r},\vec{r'}) \frac{\partial \vec{E}(\vec{r'})}{\partial n(\vec{r'})} - \frac{\partial G_0(\vec{r},\vec{r'})}{\partial n(\vec{r'})} \vec{E}(\vec{r'}) \right) + \hat{t}(\vec{r}) \cdot \nabla\phi(\vec{r}), \quad \vec{r} \in S_{nc}. \tag{2}$$

$$\phi(\vec{r}) = \int_S dS' \frac{\rho(\vec{r'})}{\varepsilon} G_0(\vec{r},\vec{r'}), \quad \vec{r} \in S. \tag{3}$$

$$\nabla \cdot \vec{E}(\vec{r}) = 0, \quad \vec{r} \in S_{nc} \tag{4}$$

$$\hat{n}(\vec{r}) \cdot \vec{E}(\vec{r}) = \frac{j\omega\rho(\vec{r})}{\sigma}, \quad \vec{r} \in S_{nc} \tag{5}$$

$$\vec{E}_t(\vec{r}) = \hat{t}(\vec{r}) \cdot \vec{E}(\vec{r}) = 0, \quad \vec{r} \in S_c \tag{6}$$

$$\hat{n}(\vec{r}) \cdot \frac{\partial \vec{E}(\vec{r})}{\partial n(\vec{r})} = 0, \quad \vec{r} \in S_c \tag{7}$$

$$\phi(\vec{r}) = constant, \quad \vec{r} \in S_c \tag{8}$$

where

$$G_0(\vec{r},\vec{r'}) = \frac{e^{jk_0|\vec{r}-\vec{r'}|}}{4\pi|\vec{r}-\vec{r'}|},\ k_0 = \omega\sqrt{\varepsilon\mu}\ , \qquad (9)$$

and

$$G_1(\vec{r},\vec{r'}) = \frac{e^{jk_1|\vec{r}-\vec{r'}|}}{4\pi|\vec{r}-\vec{r'}|},\ k_1 = -\sqrt{\omega^2\varepsilon\mu - j\omega\mu\sigma_i}\ , \qquad (10)$$

σ_i is the conductivity of the ith conductor, S_i is the surface of the ith conductor and S is the union of all conductor surfaces, and S_c and S_{nc} are the contact part and the non-contact part of the surface S, respectively. Here $\hat{t}(\vec{r})$ is a pair of vectors $\hat{t}_1$ and $\hat{t}_2$ which represent the local two dimensional coordinate system at point $\vec{r}$ on the surface.

2.2 Discretization of the formulation

In order to discretize the integral equations (1), (2) and (3), a piecewise constant centroid collocation scheme is used. The conductor surface is discretized into many flat quadrilateral panels. Seven unknowns are associated with each panel: E_x, E_y, E_z, $\frac{\partial Ex}{\partial n}$, $\frac{\partial Ey}{\partial n}$, $\frac{\partial Ez}{\partial n}$ and ρ. The scalar potential ϕ is associated with the panel vertexes, and $\nabla\phi$ in (2) is computed using finite-differences. With this setting, equations (5), (6), (7), and (8) become simple algebraic equations. But equation (4) deserves more attention.

Applying the integral form of equation (4) to the surface of an infinitely thin small rectangular box beneath the conductor surface, we obtain

$$\int_C d\gamma \vec{E}_t(\gamma)\cdot(\hat{n}(\gamma)\times\hat{l}(\gamma)) - \int_a dS(\vec{r})\hat{n}(\vec{r})\cdot\frac{\partial\vec{E}(\vec{r})}{\partial n(\vec{r})} = 0 \qquad (11)$$

where a is the top of the box, C is the periphery of a, $\hat{n}$ is the normal unit vector and $\hat{l}$ is the unit vector along C. Equation (11) is enforced on the so-called dual panel around each vertex, one dual panel $Q_1Q_2Q_3Q_4$ is shown in figure 1.

Now we can write the system matrix as (12), where the horizontal lines are used to mark the corresponding relation between row blocks and the equations (1) to (8). For example, the three rows in the first row block correspond to (1). Matrix I is the identity matrix, S_0 and S_1 are respectively the dense matrices corresponding to the single-layer integral with Green's function G_0 in (9) and G_1 in (10), D_0 and D_1 are respectively the dense matrices corresponding to the double-layer integral with Green's function G_0 and G_1, g_1 and g_2 are sparse matrices representing the finite-difference approximation of $\nabla\phi$. Sparse matrices $T_{1,\alpha}$, $T_{2,\alpha}$ and N_α ($\alpha = x,y,z$) are the transfer matrices relating the local coordinate system (t_1, t_2 and n) to the global coordinate system (x,y and z). The nonzero elements of the sparse matrices A_x, A_y and A_z and the nonzero elements of the sparse matrices C_x, C_y and C_z are related to the dual panel discretization. And ϕ_c, the known potential on the contact, is used as the excitation.

3. SCALING AND PRECONDITIONING

3.1 Scaling

The system in (12) will be solved iteratively, so keeping the condition number small is essential. Suppose the average panel edge length is $O(u)$, we will first estimate the scale of each matrix block in (12) in terms of u.

The elements of matrix S_0 in (12) are

$$S_0(i,j) = \int_{Panel_j} ds(\vec{r}')G(\vec{r}_i,\vec{r}') = \int_{Panel_j} ds(\vec{r}')\frac{e^{jk_0|\vec{r}_i-\vec{r}'|}}{4\pi|\vec{r}_i-\vec{r}'|}.$$

Since $\vec{r}_i$ and $\vec{r}'$ are on the conductor surface, it is clear that $\frac{e^{jk_0|\vec{r}_i-\vec{r}'|}}{|\vec{r}_i-\vec{r}'|}$ is $O(1/u)$ and $ds(\vec{r}')$ is $O(u^2)$. Hence $S_0(i,j)$ is $O(u)$. Obviously,

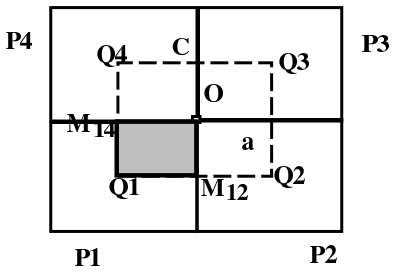

Figure 1: Dual panel

the same holds true for the elements in matrix S_1. The elements of matrix D_0 are

$$D_0(i,j) = \begin{cases} -\int_{Panel_j} ds(\vec{r}')\frac{\partial}{\partial n(\vec{r}')}\frac{e^{jk_0|\vec{r}_i-\vec{r}'|}}{4\pi|\vec{r}_i-\vec{r}'|} & i\neq j \\ -\frac{1}{2} & i=j. \end{cases}$$

Follow the same reasoning, we find that $D_0(i,j)$ is $O(1)$. Again, the same observation holds true for matrix D_1.

The dual panel discretization in (11) implies that the elements in matrices C_x, C_y and C_z are $O(u)$ and the elements in matrices A_x A_y and A_z are $O(u^2)$. And it is easy to check that the elements in the finite difference matrices g_1 and g_2 are $O(1/u)$.

Now it is clear that the scale in different matrix blocks in (12) could be different by many orders of magnitude if u is small. The huge difference in the scale could lead to large condition number. For example, the condition number could be as large as 10^{20} for micron feature sizes. Large condition numbers usually leads to inaccurate results and stagnant iterations of an iterative solver.

Fortunately, a simple scaling manipulation as the following can be used to remedy the situation: scale the first three columns and the last column with $1/u$ and the seventh column with u, and then scale the seventh row with $1/u$. It is easy to check that now most of the matrix blocks are $O(1)$. Hence the new system matrix is much better conditioned.

A simple straight wire example could be used to show the impact of the scaling. The size of the wire is $1\times1\times4um$, its conductivity is 5.8×10^7. The DC resistance of this wire should be 0.068965Ω. However, without the scaling trick presented here, the calculated resistance would be 0.0810Ω. After the system matrix has been properly scaled, the result becomes 0.068965Ω.

3.2 Preconditioning

A straightforward way of constructing a preconditioner for the system matrix like the one in (12) is to simply replace the matrix blocks corresponding to the integral operators with their diagonal elements and keep all other sparse matrix blocks. This method was used in [9] to construct a preconditioner from the system matrix in (12). Extensive numerical experiments have shown that this preconditioner could significantly reduce the number of iterations. But for some structures the number of nonzeros in the preconditioner after the sparse LU factorization is still rather large. This is partially because some rows in the preconditioner before the LU factorization are not sparse enough.

Among the unknowns of the surface formulation, $\vec{E}$ and $\frac{\partial\vec{E}}{\partial n}$ are vector variables. They could be defined either in the global coordinate system ($\hat{x}$, $\hat{y}$, $\hat{z}$) or in the local coordinate system ($\hat{t}_1$, $\hat{t}_2$, $\hat{n}$). On the other hand, vector equation (1) could also be enforced either in the global or the local coordinate system. These different options result in different system matrices. But they are in fact different by just a similarity transformation. Hence they all have the same condition number and lead to the same convergence behavior if an iterative solver is used. But the preconditioners constructed from these different system matrices are significantly different, particularly in the matrix sparsity.

$$\left[\begin{array}{ccc|ccc|c|c} S_1 & 0 & 0 & D_1 & 0 & 0 & 0 & 0 \\ 0 & S_1 & 0 & 0 & D_1 & 0 & 0 & 0 \\ 0 & 0 & S_1 & 0 & 0 & D_1 & 0 & 0 \\ \hline T_{1,x}S_0 & T_{1,y}S_0 & T_{1,z}S_0 & T_{1,x}D_0 & T_{1,y}D_0 & T_{1,z}D_0 & g_1 & 0 \\ T_{2,x}S_0 & T_{2,y}S_0 & T_{2,z}S_0 & T_{2,x}D_0 & T_{2,y}D_0 & T_{2,z}D_0 & g_2 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & -I & S_0 \\ \hline -A_x & -A_y & -A_z & C_x & C_y & C_z & 0 & 0 \\ \hline 0 & 0 & 0 & N_x & N_y & N_z & 0 & \frac{-j\omega}{\sigma} I \\ \hline 0 & 0 & 0 & T_{1,x} & T_{1,y} & T_{1,z} & 0 & 0 \\ 0 & 0 & 0 & T_{2,x} & T_{2,y} & T_{2,z} & 0 & 0 \\ \hline N_x & N_y & N_z & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & I & 0 \end{array}\right] \left[\begin{array}{c} \frac{\partial E_x}{\partial n} \\ \frac{\partial E_y}{\partial n} \\ \frac{\partial E_z}{\partial n} \\ \hline E_x \\ E_y \\ E_z \\ \hline \phi \\ \hline \rho \end{array}\right] = \left[\begin{array}{c} 0 \\ 0 \\ 0 \\ \hline 0 \\ 0 \\ \hline 0 \\ \hline 0 \\ \hline 0 \\ \hline 0 \\ 0 \\ \hline 0 \\ \hline \phi_c \end{array}\right] \tag{12}$$

Table 1: Performance of preconditioners in the global and the local coordinate system

	local	global
number of nonzeros before LU	320376	547636
number of nonzeros after LU	1097973	1318222
number of fill-in's	777597	770586
number of GMRES iterations	15	15

A sparser precondioner can be derived by defining $\vec{E}$ and $\frac{\partial \vec{E}}{\partial n}$ as well as enforcing equation (1) in the local coordinate system. The structure of the system matrix constructed under such condition is shown in (13).

$$\left[\begin{array}{ccc|ccc|c|c} T_{11}S_1 & T_{12}S_1 & T_{13}S_1 & T_{11}D_1 & T_{12}D_1 & T_{13}D_1 & 0 & 0 \\ T_{21}S_1 & T_{22}S_1 & T_{23}S_1 & T_{21}D_1 & T_{22}D_1 & T_{13}D_1 & 0 & 0 \\ T_{31}S_1 & T_{32}S_1 & T_{33}S_1 & T_{31}D_1 & T_{32}D_1 & T_{33}D_1 & 0 & 0 \\ \hline T_{11}S_0 & T_{12}S_0 & T_{13}S_0 & T_{11}D_0 & T_{12}D_0 & T_{13}D_0 & g1 & 0 \\ T_{21}S_0 & T_{22}S_0 & T_{23}S_0 & T_{21}D_0 & T_{22}D_0 & T_{23}D_0 & g2 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & -I & S_0 \\ \hline 0 & 0 & -A & C_{t_1} & C_{t_2} & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & I & 0 & \frac{-j\omega}{\sigma} I \\ \hline 0 & 0 & 0 & I & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & I & 0 & 0 & 0 \\ \hline 0 & 0 & I & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & I & 0 \end{array}\right] \tag{13}$$

Matrices $T_{mn}(m,n=1,2,3))$ are defined as $T_{11}(i,j)=\hat{t}_1^{(i)}\cdot\hat{t}_1^{(j)}$, $T_{12}(i,j)=\hat{t}_1^{(i)}\cdot\hat{t}_2^{(j)}$, $T_{13}(i,j)=\hat{t}_1^{(i)}\cdot\hat{n}^{(j)}$, and etc, where $(\hat{t}_1^{(i)},\hat{t}_2^{(i)},\hat{n}^{(i)})$ is the local coordinate system on the ith panel. Extracting the diagonal part of the matrix blocks that correspond to the integral operators and keeping the remaining sparse matrices yields a sparser preconditioner. The density of all rows below the third block row have been reduced by about one half.

To verify the effectiveness of the sparser preconditioner, we used it in the analysis of a four-turn spiral over a lossy substrate ground plane. The total number of unknowns is 72531. The performance of the preconditioners in the global and the local coordinate system is compared in table 1. As it is expected, the preconditioner in the local coordinate system before the LU factorization is indeed much sparser. But this advantage is somewhat offset by the fill-in's generated by LU factorization. Hence the number of nonzeros in both precondioners after LU factorization is different by about 20%. This directly translates into a 20% saving in memory usage. In addition, both preconditioners lead to the same iteration count.

4. PRE-CORRECTED FFT ALGORITHM

After discretization, the algebraic equations (5), (6), (7), (8) and (11) become sparse matrix equations. But integral equations (1), (2) and (3) become dense matrix equations. So solving the whole system matrix using iterative methods still takes $O(N^2)$ operations, where N is the number of unknowns. In this paper, we use the pre-corrected FFT algorithm to accelerate the dense matrix vector product corresponding to the operation of those integral operators in (1), (2) and (3).

An abstract form of the kernels in (1), (2) and (3) is

$$K(\vec{r'},\vec{r}) = \mathcal{F}_1(\mathcal{F}_2(G(\vec{r'},\vec{r}))) \tag{14}$$

where $G(\vec{r'},\vec{r})$ is the Green's function, and the most commonly used forms of the operator $\mathcal{F}_1(\cdot)$ and $\mathcal{F}_2(\cdot)$ are $\mathcal{F}_1(\cdot)=U(\cdot)$, $\frac{d(\cdot)}{dx(\vec{r})}$, $\frac{d(\cdot)}{dy(\vec{r})}$, $\frac{d(\cdot)}{dz(\vec{r})}$, $\frac{d(\cdot)}{dn(\vec{r})}$ and $\mathcal{F}_2(\cdot)=U(\cdot)$, $\frac{d(\cdot)}{dx(\vec{r'})}$, $\frac{d(\cdot)}{dy(\vec{r'})}$, $\frac{d(\cdot)}{dz(\vec{r'})}$, $\frac{d(\cdot)}{dn(\vec{r'})}$, and $U(\cdot)$ is the identity operator. For the sake of clarity, we use a simple single-kernel integral equation

$$\int_S dS' K(\vec{r'},\vec{r})\rho(\vec{r'}) = f(\vec{r}),\ \vec{r}\in S \tag{15}$$

to illustrate how the pFFT algorithm can be used to accelerate the operation of an integral operator. Function $f(\vec{r})$ is the known right hand side term. The procedure extends easily to the integral equations with multiple kernels, such as (1) and (2).

The standard procedure for solving equation (15) numerically is to discretize by means of projection [7] and solve the resulting linear system with an iterative method, such as GMRES [12]. For example, if $\rho(\vec{r})$ is approximated as a weighted combination of N basis functions, $\rho(\vec{r})=\sum\alpha_j b_j(\vec{r})$, and α_j are determined by collocation method, then we arrive at

$$[A]\overline{\alpha} = \overline{f}, \tag{16}$$

where

$$A_{i,j} = \int_{\Delta_j^b} dS' K(\vec{r'},\vec{r}_i) b_j(\vec{r'}), \tag{17}$$

$\vec{r}_i$ is the ith collocation point, and Δ_j^b is the support of the basis functions $b_j(\vec{r})$. The commonly used basis functions are low-order polynomials with local support [7].

The key idea of the pFFT algorithm is to replace (16) with its sparse representation

$$[A]\overline{\alpha} = ([D]+[I][H][P])\overline{\alpha}, \tag{18}$$

where I, P, D and H are the interpolation, projection, direct and convolution matrix, respectively. Algorithms used in this paper to compute matrices D and H are similar to the ones in [8]. Hence we shall not give details here. The contribution of this paper is to use a different way than the one in [8] to set up matrices I and P so that it could easily handle the general kernels in (14).

4.1 Projection and interpolation

Figure 2 shows a 2D pictorial representation of the projection step, where a triangle is used to represent the support of the basis function. A 3×3 projection grid is assumed here and obviously more points could be used if the accuracy requirement is higher.

We start with a point charge ρ_p at point **S** on the triangle, shown in figure 2. The potential at point **E** due to this point charge is

$$\phi_E^{(1)} = \rho_p G(\vec{r}_s, \vec{r}_E). \tag{19}$$

The purpose of the projection is to find a set of grid charges $\bar{\rho}_g$ on the projection grid points such that they generate the same potential at point **E**, i.e.,

$$\phi_E^{(2)} = \sum_i \rho_{g,i} G(\vec{r}_i, \vec{r}_E) = (\bar{\rho}_g)^t \bar{\phi}_g = \phi_E^{(1)} \tag{20}$$

where $\phi_{g,i} = G(\vec{r}_i, \vec{r}_E)$. We could use a set of polynomials to expand the Green's function

$$G(\vec{r}, \vec{r}_E) = \sum_k f_k(\vec{r}) c_k = \bar{f}^t(\vec{r}) \bar{c}. \tag{21}$$

A simple set of polynomial functions for this purpose is $f_k(\vec{r}) = f_k(x,y) = x^i y^j, i, j = 0, 1, 2, k = 2i + j$. Matching both sides at each grid point $\vec{r}_i$ yields a linear system $[F]\bar{c} = \bar{\phi}_g$, where the i-th row of the matrix $[F]$ is the set of polynomials $\bar{f}(\vec{r})$ evaluated at point r_i. Substituting the solution $\bar{c} = F^{-1}\bar{\phi}_g$ into (21) and evaluating it at point **S** yields

$$G(\vec{r}_s, \vec{r}_E) = \bar{f}^t(\vec{r}_s) F^{-1} \bar{\phi}_g. \tag{22}$$

In light of (19) and (20) we have

$$(\bar{\rho}_g)^t = \rho_p \bar{f}^t(\vec{r}_s) F^{-1}, \tag{23}$$

the projection charges for a point charge.

A charge distribution $b_j(\vec{r})$ on the jth basis function support could be regarded as a linear combination of an infinite number of point charges. Equation (23) implies that the projection charges are linearly proportional to the point charge, hence it easily follows that the projection charges for the charge distribution $b_j(\vec{r})$ is

$$(\bar{\rho}_g^{(j)})^t = \left(\int_{\Delta_j^b} dS b_j(\vec{r}) \bar{f}^t(\vec{r}) \right) [F]^{-1}. \tag{24}$$

If there are more than one basis function in the approximate solution, then the total grid charges Q_g on each grid point is the accumulation of the grid charge due to each basis function. It could be written as

$$\bar{Q}_g = \sum_{j=1}^{N_b} \alpha_j \bar{\rho}_g^{(j)} = [P]\bar{\alpha}, \tag{25}$$

where $[P]$ is an $N_g \times N_b$ matrix, N_b is the number of basis functions, N_g is the number of grid points, and the magnitude of basis functions $\bar{\alpha}$ is defined in (16). Due to the locality of the basis support, the projection grid for each basis function has only a small number of points. Hence each column of the projection matrix $[P]$ is rather sparse. The non-zero elements in the j-th column of matrix $[P]$ are the elements of the column vector $\bar{\rho}_g^{(j)}$ in equation (24).

If the kernel has a differential operator inside the integral, the potential at point **E** due to a point charge is

$$\phi_E^{(1)} = \frac{\partial}{\partial \beta(\vec{r}_s)} [\rho_p G(\vec{r}_s, \vec{r}_E)] = \frac{\partial}{\partial \beta(\vec{r}_s)} [\rho_p \bar{f}^t(\vec{r}_s) F^{-1} \bar{\phi}_g]. \tag{26}$$

where β stands for x, y or n. We again want to find a set of grid charges $\bar{\sigma}_\beta$ on the projection grid points such that they generate the same potential at point **E**, i.e.,

$$\phi_E^{(2)} = \sum_i \sigma_{\beta,i} G(\vec{r}_i, \vec{r}_E) = (\bar{\sigma}_\beta)^t \bar{\phi}_g = \phi_E^{(1)}. \tag{27}$$

Equations (26) and (27) imply that the projection charges are

$$(\bar{\sigma}_\beta)^t = \frac{\partial}{\partial \beta(\vec{r}_s)} \left(\rho_p \bar{f}^t(\vec{r}_s) F^{-1} \right). \tag{28}$$

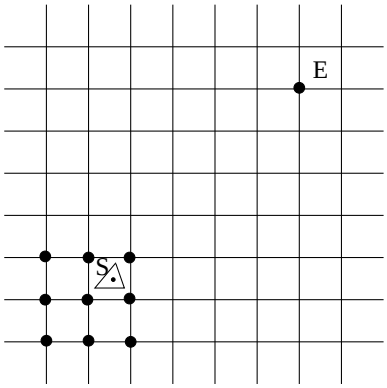

Figure 2: 2-D pictorial representation of the projection step

Similar to the single-layer operator case, the projection charges for a charge distribution $b_j(\vec{r})$ on the jth basis function support is

$$(\bar{\sigma}_\beta^{(j)})^t = \left(\int_{\Delta_j^b} dS b_j(\vec{r}) \frac{\partial}{\partial \beta(\vec{r})} \bar{f}^t(\vec{r}) \right) [F]^{-1}. \tag{29}$$

The projection matrix for the kernel with a differential operator is structurally identical to the matrix $[P]$ in equation (25). The non-zero elements in the j-th column of the matrix are the elements of the column vector $\bar{\sigma}_\beta^{(j)}$ in equation (29).

It is shown in [13] that the interpolation matrix I has a dual relationship to the projection matrix. Hence the algorithm to form matrix I is very similar to the one shown in this section. Please refer to [14] for details.

4.2 Summary

Since polynomials are used in both the interpolation and the projection, the interpolation matrix I and the projection matrix P are only related to the operator $\mathcal{F}_1$ and $\mathcal{F}_2$, respectively, not to the Green's function $G(\vec{r}, \vec{r'})$. This makes it much easier to handle the complicated kernels $K(\vec{r'}, \vec{r})$ in (14). In addition, we could exploit the fact that matrices I and P are not related to Green's function, hence frequency-independent [5]. For example, we could form these two matrices just once and use them for Helmholtz kernel at various frequencies.

5. NUMERICAL RESULTS

Base upon the algorithm described in section 4, we have developed a C++ program called pfft++. Combining pfft++ with the surface formulation, we have developed FastImp, a fast impedance extraction program. In this section, we first use small examples to demonstrate FastImp's accuracy. We then use a few large examples to demonstrate FastImp's speed and capacity.

5.1 Validation of FastImp

Magneto-Quasi-Static (MQS) analysis. A ring example is used to verify the accuracy of the MQS analysis by FastImp. We also intend to use this relatively small example to conduct the convergence test of FastImp. Because of limited space, we only show inductance results here. The resistance results are essentially the same as those shown in [11]. The ring is 10*mm* in radius, with a square cross section of the size 0.5*mm* by 0.5*mm*. The conductivity is that of the copper, which is 5.8e7. The low frequency inductance calculated using the formula in [15] is 48.89 nH. The number of filaments used by FastHenry is 960, 3840 and 15360, respectively. And the number of panels used by FastImp is 992 and 2048, respectively. With refinement of panel discretization, FastImp's results converges very well, as shown in figure 4.

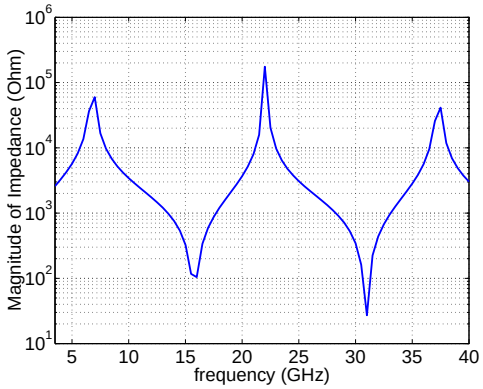

Figure 3: Impedance of a shorted transmission line

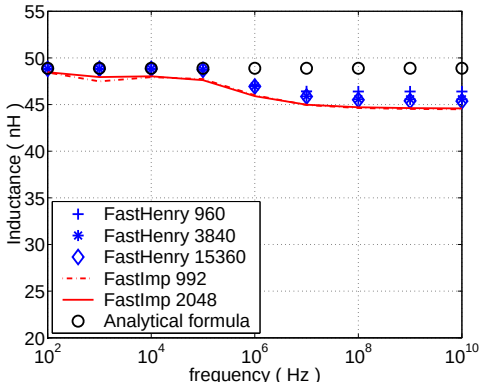

Figure 4: Inductance of a ring

Electro-Magneto-Quasi-Static (EMQS) analysis. We have used FastImp to carry out the EMQS analysis of a shorted transmission line. The length of the transmission line is 1*cm*. The cross-section of each conductor is $50 \times 50um$. And the space between two conductors is 50*um*. The first three resonance frequencies should be 7.5GHz, 22.5GHz and 37.5GHz. Figure 3 shows the magnitude of its impedance at different frequency points, the expected resonance frequencies are clearly shown in the plot. This suggests that FastImp could accurately capture both the inductive and the capacitive effects. This shorted transmission line example validates the EMQS analysis of FastImp.

5.2 Performance of FastImp

A few large and practical structures are analyzed in this section. The CPU time and memory usage for different examples are compared in table 2.

Multiple conductor crossover bus. Figure 5 shows a multiple conductor bus with three-layer of identical conductors. Each layer has 10 conductors and the conductors on different layer are orthogonal to each other. The size of every conductor is $1 \times 1 \times 25$um. We only extracted one column of the impedance matrix (since this is a multiple port structure) at one frequency point $f = 1$GHz using the EMQS analysis. The CPU time and memory usage are shown in table 2.

Stacked spirals over ground. The impedance matrix of stacked two 9-turn circular spirals over a lossy ground plane and two stacked 8-turn rectangular spirals over a lossy ground plane are extracted at one frequency point $f = 1$GHz using the EMQS analysis. The CPU time and memory usage are shown in table 2.

Large 3D structures. FastImp has been used to perform the EMQS analysis of two large structures shown in figure 6 and 7. The discretization, detailed breakdown of CPU time and memory usage for the analysis of these two examples is shown in table 3, 4 and 5, respectively. The computation was carried out on a server with 32Gb memory and one 64-bit Itanium micro-processor. This server is about 3 times slower than the desktop computer used for the previous examples.

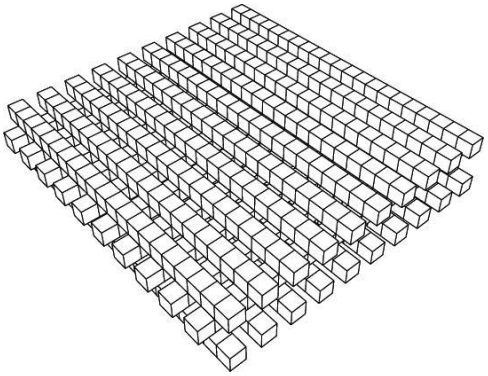

Figure 5: Multiple conductor bus

Table 2: Comparison of CPU time and memory usage for various practical structures. The calculations were carried out on a desk top computer with a pentium III micro-processor and 1Gb memory.

	bus	circular spirals	rectangle spirals
#panels	18,540	15,194	18,520
#unknowns	148,380	121,558	148,166
FastImp	9min,340Mb	68min,642Mb	54min,749Mb
*iterative	160min,19Gb	750min,72Gb	590min,83Gb
*standard	136days,19Gb	100days,19Gb	168days,22Gb

*obtained by estimation or extrapolation.

Computational complexity of FastImp. We have used FastImp to analyze a series of similar structures with increasingly larger size. These structures are 1x1, 2x2, 4x4 and 8x8 spiral arrays. All elements in these arrays are 3-turn rectangular spirals. The CPU time versus number of spiral elements in the spiral arrays is shown in figure 8. The plot clearly indicates that the CPU time grows nearly linearly with the problem size.

6. CONCLUSIONS

We have generalized the pre-corrected FFT algorithm to allow the acceleration of complicated integral operators. Based on this generalization we have developed a flexible and extensible fast integral equation solver, pfft++, whose computational complexity is nearly $O(N)$. Using pfft++ as the engine, we have developed a fast impedance extraction program, FastImp. Numerical examples show that FastImp can perform MQS and EMQS analysis of 3D general structures across wide frequency range, from zero frequency to at least hundreds of giga hertz. It takes FastImp anywhere between 10 minutes to about one hour to analyze realistic 3D structures with many hundred thousand of unknowns. Both pfft++ and FastImp are now available at rleweb.mit.edu/vlsi/codes.htm.

Table 3: Discretization of the MIT logo example and the 16x8 spiral array example

	MIT logo	16x8 spiral array
number of panels	173,184	180,224
number of unknowns	1,385,718	1,442,048
number of grids	$1024 \times 256 \times 8$	$256 \times 128 \times 8$
grid step size	1.89um	1.91um

Table 4: A detailed breakdown of the CPU time used by the MIT logo example and the 16x8 spiral array example. Unit is second

	MIT logo	16x8 array
P and I matrices	890	849
D and H matrices	13638	14353
form the preconditioner P_r	54	53
LU factorization of P_r	1512	1927
GMRES (tol = 1e-3)	32424 (77 iter)	25168 (80 iter)
total	51244	42350

Table 5: A detailed breakdown of the memory usage for the MIT logo example and the 16x8 spiral array example. Unit is Gb

	MIT logo	16x8 spiral array
direct matrices	5.17	5.54
projection matrices	0.38	0.39
interpolation matrices	0.22	0.23
convolution matrices	0.68	0.13
maps between grids and panels	0.65	0.70
preconditioner	2.72	2.76
GMRES	2.03	2.21
total	11.85	11.96

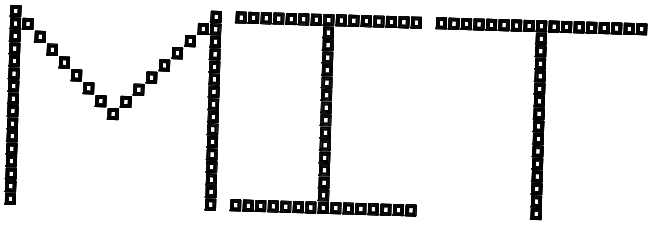

Figure 6: A large MIT logo consisting of 123 3-turn square spirals

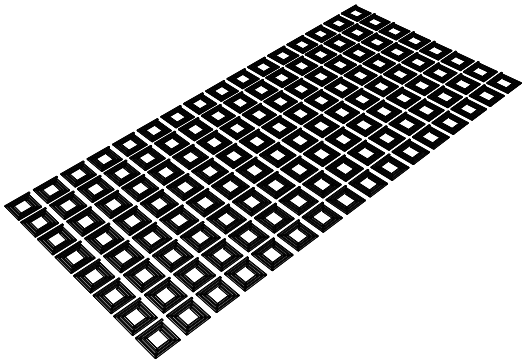

Figure 7: 16x8 3-turn rectangular spiral array

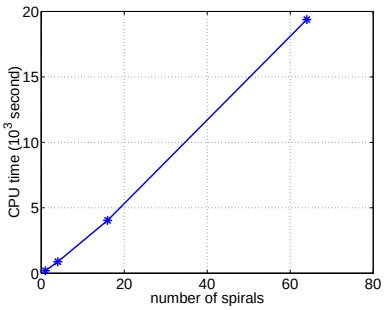

Figure 8: The CPU time vs. number of spirals in the spiral arrays

7. REFERENCES

[1] K. Nabors and J. White, "FASTCAP: A multipole-accelerated 3-D capacitance extraction program," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 10, pp. 1447–1459, November 1991.

[2] M. Kamon, M. J. Tsuk, and J.K. White, "FastHenry: A multipole-accelerated 3-D inductance extraction program," *IEEE Transactions on Microwave Theory and Techniques*, vol. 42, no. 9, pp. 1750–1758, September 1994.

[3] S. Kapur and D.E. Long, "IES3: A fast integral equation solver for efficient 3-dimensional extraction," *International Conference on Computer Aided-Design*, pp. 448–455, 1997.

[4] M. Bachtold, M. Spasojevic, C. Lage, and P.B. Ljung, "A system for full-chip and critical net parasitic extraction for ulsi interconnects using a fast 3-d field solver," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 19, no. 3, pp. 325–338, 2000.

[5] Joel R. Phillips, Eli Chiprout, and David D. Ling, "Efficient full-wave electromagnetic analysis via model-order reduction of fast integral transformations," *ACM/IEEE Design Automation Conference*, 1995.

[6] L. Greengard and V. Rohklin, "A fast algorithm for particle simulations," *Journal of Computational Physics*, vol. 73, no. 2, pp. 325–348, December 1987.

[7] Wolfgang Hackbush, *Integral Equations, Theory and Numerial Treatment*, Birkhauser Verlag, Basel, Switzerland, 1989.

[8] Joel R. Phillips and J. K. White, "A precorrected-FFT method for electrostatic analysis of complicated 3D structures," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pp. 1059–1072, 1997.

[9] J. Wang and J. K. White, "A wide frequency range surface integral formualtion for 3D RLC extraction," *International Conference on Computer Aided-Design*, 1999.

[10] Junfeng Wang, *A new surface integral formulation of EMQS impedance extraction for 3-D structures*, Ph.D. thesis MIT EECS Department, 1999.

[11] Zhenhai Zhu, Jingfang Huang, Ben Song, and J. K. White, "Improving the robustness of a surface integral formulation for wideband impendance extraction of 3D structures," *International Conference on Computer Aided-Design*, pp. 592–597, 2001.

[12] Youcef Saad and Martin Schultz, "GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems," *SIAM J. Sci. Statist. Comput.*, vol. 7, no. 3, pp. 856–869, July 1986.

[13] Joel R. Phillips, *Rapid solution of potential integral equations in complicated 3-dimensional geometries*, Ph.D. thesis MIT EECS Department, 1997.

[14] Zhenhai Zhu, *Efficient Techniques for Wideband Impedance Extraction of Complex 3-dimensional Geometries*, Master thesis MIT EECS Department, 2002.

[15] Frederick Warren Grover, *Inductance calculations, working formulas and tables*, Govt. Print. Off., New York, NY, 1946.