A Direct Finite Element Solver of Linear Complexity for Large-Scale 3-D Circuit Extraction in Multiple Dielectrics

Bangda Zhou, Haixin Liu, and Dan Jiao
School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47907

ABSTRACT

We develop a direct finite-element solver of linear (optimal) complexity to extract broadband circuit parameters such as S-parameters of arbitrarily shaped 3-D interconnects in inhomogeneous dielectrics. Numerical experiments demonstrate a clear advantage of the proposed solver as compared with existing finite-element solvers that employ state-of-the-art direct sparse matrix solutions. A linear complexity in both CPU time and memory consumption is achieved with prescribed accuracy satisfied. A finite-element matrix from the analysis of a large-scale 3-D circuit in multiple dielectrics having 5.643 million unknowns is directly factorized in less than 2 hours on a single core running at 2.8 GHz.

Categories and Subject Descriptors

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

General Terms

Algorithms

Keywords

Circuit extraction, interconnect, finite element methods, direct solvers

1. INTRODUCTION

Multicore and many-core computing have become a new form of equivalent scaling to accompany the continuation of Moore's Law. As information needs to be transferred in and out of the processors at a throughput proportional to the computational performance, a concurrent rapid scaling of processor input/output (I/O) bandwidth is required. Thus, the design of high-bandwidth I/O has become a new challenge in current and future integrated circuit and system design. To meet such a challenge, accurate and efficient modeling of non-quasi-static effects, substrate noise, high-frequency noise, and parasitic coupling is called for. Modeling of effects that have a more global influence such as cross talk, substrate return path, substrate coupling, and electromagnetic radiation is also demanded. The advent of new technologies such as 3D integration by the use of TSVs and wireless signaling with passive devices further challenges the traditional quasi-statics and/or 2D based circuit modeling approaches.

The fullwave based methods for 3-D circuit modeling can be categorized into two broad classes: integral equation (IE) based methods and partial differential equation based methods. In the latter, a representative method is the finite element method (FEM). A surface IE-based method reduces a 3-D volumetric problem to a surface problem. However, its formulation becomes cumbersome when the circuit to be extracted involves complicated materials. Little work has been reported in a surface IE-based fullwave extraction of 3-D lossy circuits in nonuniform dielectrics. A volume IE-based method is capable of handling arbitrary nonuniform materials with great ease, however, the resulting linear system of equations is not only dense but also large involving volume unknowns in the entire 3-D circuit. Therefore, when the problem of interest involves complicated materials, the FEM method has been a popular method for choice because of its great capability in handling both irregular geometries and inhomogeneous materials.

As a partial differential equation based method, the FEM produces a sparse system matrix for solving Maxwell's equations. Although the system matrix is sparse, its inverse as well as LU factors are generally dense. As a result, solving it can be a computational challenge when the matrix size is large. A traditional direct solution is computationally expensive. It is shown in [1] that the optimal operation count of the direct solution of an FEM matrix in exact arithmetic is $O(N^{1.5})$ for 2-D problems, and $O(N^2)$ for 3-D problems, where N is the matrix dimension. Although there have been successes in speeding up the direct finite element solution by a multifrontal based algorithm [2] or by an $\mathcal{H}$-matrix based mathematical framework [3] for 3D fullwave analysis, as yet, no $O(N)$ complexity, i.e. optimal complexity, has been accomplished for the FEM-based direct solution of general 3-D circuit problems.

State-of-the-art fast FEM-based solvers rely on iterative approaches to solve large-scale circuit problems. The computational complexity of an iterative solver is $O(N_{it} N_{rhs} N)$ at best, where N_{it} is the number of iterations, and N_{rhs} the number of right hand sides. When N_{it} and N_{rhs} are large,

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
DAC '13, May 29 - June 07 2013, Austin, TX, USA.
Copyright 2013 ACM 978-1-4503-2071-9/13/05 ...$15.00.

state-of-the-art iterative solutions become inefficient since the entire iteration procedure has to be repeated for each right hand side. To give an example, assuming the CPU cost for each port is 30 minutes, to assess crosstalk between 64 ports in a high-speed I/O, one has to wait for 1.3 days. Furthermore, the complexity of an iterative solver is problem dependent since the iteration number N_{it} is, in general, problem dependent.

The contribution of this paper is a direct FEM solver of linear complexity for extracting fullwave circuit parameters from arbitrarily shaped 3-D lossy conductors in inhomogeneous materials. Both theoretical analysis and numerical experiments have demonstrated its linear complexity in both CPU time and memory consumption with prescribed accuracy satisfied. The proposed direct solver successfully factorizes an FEM matrix having 5,643,240 unknowns resulting from the analysis of a large-scale 3-D lossy circuit in multiple dielectrics in less than 2 hours on a single core running at 2.8 GHz. Comparisons with state-of-the-art direct FEM solvers that employ the most advanced direct sparse matrix solutions as well as a commercial iterative FEM solver have shown a clear advantage of the proposed direct solver.

2. VECTOR FINITE-ELEMENT ANALYSIS

Consider a general 3-D circuit with arbitrarily shaped lossy conductors and inhomogeneous dielectrics that can be lossless, lossy, and dispersive. The physical phenomena in such a circuit are governed by the second-order vector wave equation

$$\nabla \times (\frac{1}{\mu_r} \times \mathbf{E}) + jk_0\eta_0\sigma\mathbf{E} - k_0^2\varepsilon_r\mathbf{E} = -jk_0Z_0\mathbf{J}, \quad (1)$$

where $\mathbf{E}$ is electric field intensity, μ_r is relative permeability, ε_r is relative permittivity, σ is conductivity, k_0 is free-space wave number, Z_0 is free-space wave impedance, and $\mathbf{J}$ is current density. A finite element based solution of (1) results in the following linear system of equation:

$$\mathbf{Y}\{u\} = \{b\}, \quad (2)$$

in which b denotes a vector of current sources, $\mathbf{Y}$ is a sparse matrix, which can be written as

$$\mathbf{Y} = -k_0^2\mathbf{T} + jk_0\mathbf{G} + \mathbf{S} \quad (3)$$

where $\mathbf{T}$, $\mathbf{G}$, and $\mathbf{S}$ are assembled from their elemental contributions as the following

$$\begin{aligned}
\mathbf{T}_{ij}^e &= \iiint_V \varepsilon_r\mathbf{N}_i \cdot \mathbf{N}_j \mathrm{d}V \\
\mathbf{S}_{ij}^e &= \iiint_V \frac{1}{\mu_r}(\nabla \times \mathbf{N}_i) \cdot (\nabla \times \mathbf{N}_j)\mathrm{d}V \quad (4)\\
\mathbf{G}_{ij}^e &= \iiint_V \eta_0\sigma\mathbf{N}_i \cdot \mathbf{N}_j \mathrm{d}V + \iint_S (\hat{n} \times \mathbf{N}_i) \cdot (\hat{n} \times \mathbf{N}_j)\mathrm{d}S,
\end{aligned}$$

where V denotes the volume of each element e, S is the outermost boundary of the entire circuit, $\hat{n}$ is a unit normal vector, and $\mathbf{N}$ is the vector basis function used to expand unknown $\mathbf{E}$ in each element.

3. MATHEMATICAL BACKGROUND

In state-of-the-art multifrontal based direct sparse solvers [2, 5], the overall factorization of a sparse matrix is organized into a sequence of partial factorizations of smaller dense frontal matrices. Various ordering techniques have been adopted to reduce the number of fill-ins introduced during the direct matrix solution process. The computational cost of a multifrontal based solver depends on the number of nonzero elements in the $\mathbf{L}$ and $\mathbf{U}$ factors of the sparse matrix. This solver is based on exact arithmetic, and hence its optimal complexity is higher than linear complexity [1].

Recently, it is proved in [3] that the sparse matrix resulting from a finite-element based analysis of electromagnetic problems can be represented by an $\mathcal{H}$ matrix [4] without any approximation, and the inverse as well as $\mathbf{L}$ and $\mathbf{U}$ of this sparse matrix has a data-sparse $\mathcal{H}$-matrix approximation with a controlled error. It is shown that an $\mathcal{H}$-matrix based direct FEM solver has a complexity of $O(NlogN)$ in storage and a complexity of $O(Nlog^2N)$ in CPU time for solving general 3-D circuit problems.

In an $\mathcal{H}$ matrix [4], the matrix blocks are partitioned into multilevel admissible blocks and inadmissible blocks. The admissible blocks appear in the off-diagonal blocks. Consider a matrix block $\mathbf{Y}^{t,s}$ formed by unknown sets t and s, if it satisfies the following admissibility condition

$$\min\{diam(\Omega_t), diam(\Omega_s)\} < \eta dist(\Omega_t, \Omega_s), \quad (5)$$

then it is admissible. In the above, Ω_t denotes the support of all basis functions belonging to set t, Ω_s denotes the support of all basis functions belonging to set s, $diam(\cdot)$ is the Euclidean diameter, $dist(\cdot, \cdot)$ is the Euclidean distance between two sets, and η is a positive parameter that can be used to control the accuracy of the admissibility condition. A matrix block that does not satisfy the above admissibility condition is called inadmissible. In an $\mathcal{H}$ matrix, an inadmissible block keeps its original full matrix form, while an admissible block is represented by a low-rank matrix shown as the following:

$$\tilde{\mathbf{Y}}^{t \times s} = \mathbf{A}_{\#t \times k} \cdot \mathbf{B}_{\#s \times k}^T, \quad (6)$$

where k is the rank which is bounded by a constant for circuit problems [3], and # denotes the cardinality of a set. The error of a rank-k approximation can be evaluated as [4]

$$\|\mathbf{Y}^{t \times s} - \tilde{\mathbf{Y}}^{t \times s}\|_2 = \sigma_{k+1}, \quad (7)$$

in which σ_{k+1} is the maximum singular value among truncated singular values. By applying (7), the error of an $\mathcal{H}$-matrix representation can be quantitatively controlled. With a rank-k representation, an $\mathcal{H}$ matrix significantly accelerates matrix-related operations and reduces storage cost for a prescribed accuracy as compared to a full-matrix based representation.

4. PROPOSED LINEAR-COMPLEXITY DIRECT FINITE ELEMENT SOLVER

In the proposed method, we fully take advantage of the zeros in the original FEM matrix, and also the zeros in the $\mathbf{L}$ and $\mathbf{U}$ factors. Instead of treating $\mathbf{L}/\mathbf{U}$ as a whole $\mathcal{H}$-matrix like what is done in [3], we only store the nonzeros in $\mathbf{L}$ and $\mathbf{U}$ with a compact error-controlled $\mathcal{H}$-matrix representation, compute these nonzeros with fast $\mathcal{H}$-matrix based arithmetic, while completely removing all the zeros in $\mathbf{L}$ and $\mathbf{U}$ from storage and computation. Since the geometry of a 3-D circuit is known, we maximize the zeros in $\mathbf{L}$ and $\mathbf{U}$ by nested dissection ordering [1]. We build an elimination tree based on the nested dissection ordering, and precisely

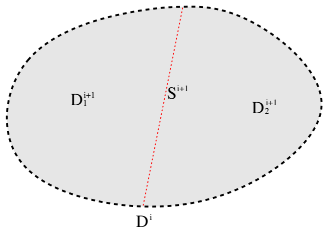

Figure 1: An example of nested dissection on domain cluster D^i

identify the nonzeros in $\mathbf{L}$ and $\mathbf{U}$ by finding a tight boundary of each node in the elimination tree. The boundary of each node is the union of the unknowns that have a crosstalk with the unknowns contained in the node, in $\mathbf{L}$ and $\mathbf{U}$. Since each nonleaf node in the elimination tree is a 2-D separator, the boundary of each node is essentially the union of the unknowns residing on the bounding box of the 2-D separator, thus the boundary size is proportional to the node size, which is 2-D. Since the computation associated with each node in the elimination tree is essentially a dense matrix operation of size (node+boundary), we reduce the factorization of the original large 3-D FEM matrix to a sequence of factorizations of 2-D dense matrices. We then develop efficient $\mathcal{H}$-matrix based algorithms to accelerate the computation of all the intermediate 2-D dense matrices. After adding the cost associated with each node in the elimination tree, the computational complexity of the proposed direct finite element solver can be theoretically proved to be $O(N)$, which will be detailed in Section 5. The overall algorithm of the proposed direct solver has six major steps:

1. Build cluster tree $T_{\mathcal{I}}$ based on nested dissection
2. Build elimination tree $E_{\mathcal{I}}$ from $T_{\mathcal{I}}$
3. Do symbolic factorization guided by $E_{\mathcal{I}}$ to obtain the boundary for each node $N^{\{j\}}$ in $E_{\mathcal{I}}$
4. Generate $\mathcal{H}$-matrix structure $\mathbf{F}^{\mathcal{H}}_{N(j)}$ for each node $N^{\{j\}}$ in $E_{\mathcal{I}}$
5. Do numerical factorization guided by $E_{\mathcal{I}}$ by $\mathcal{H}$-matrix-based algorithms
6. Solve for a right hand side based on $E_{\mathcal{I}}$. (8)

4.1 Build cluster tree $\mathbf{T}_{\mathcal{I}}$ from nested dissection

Nested dissection [1] is an ordering scheme based on geometrical information. Different ordering techniques can result in a different number of fill-ins and the nested dissection ordering has been proved to be optimal for a finite element matrix in the sense that it minimizes the number of fill-ins. With nested dissection, a computational domain is recursively divided into two separated subdomains and one interface. An example is shown in Figure 1, where at level i, a computational domain D^i is divided into three parts D_1^{i+1}, D_2^{i+1}, and S^{i+1}, in which D denotes a domain and S denotes an interface. Subdomain D_1^{i+1} and D_2^{i+1} are separated by interface S^{i+1}. Let $\mathbf{L}_{D_2^i \times D_1^i}$ be a matrix block in $\mathbf{L}$ in the D_2^i-row and D_1^i-column, and $\mathbf{U}_{D_1^i \times D_2^i}$ be a matrix block in $\mathbf{U}$ in the D_1^i-row and D_2^i-column. We can order the

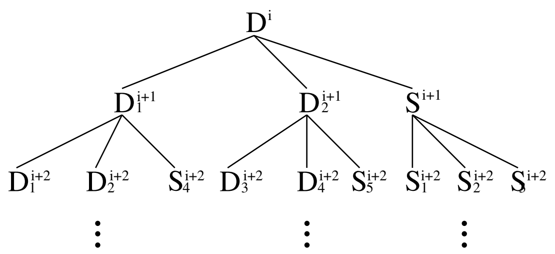

Figure 2: An example of cluster tree $T_{\mathcal{I}}^i$ of domain cluster D^i

unknowns in domain D_i as $\{D_1^i, D_2^i, S^i\}$, then the $\mathbf{L}_{D_2^i \times D_1^i}$ and $\mathbf{U}_{D_1^i \times D_2^i}$ will become zero, and hence reducing fill-ins. Moreover, if the interface S^{i+1} is also further divided, i.e., S^{i+1} is divided and ordered as S_1^{i+2}, S_2^{i+2} and S_3^{i+2}, then S_1^{i+2} and S_3^{i+2} are also completely separated by S_2^{i+2}. The recursive procedure of the nested dissection results in a tree structure shown in Figure 2. Each node of the tree is called a cluster in an $\mathcal{H}$-matrix-based representation, and the entire reversed tree that is rooted at the whole unknown set $\mathcal{I}$ and ended at the leaf unknown sets is called a cluster tree. Figure 2 shows a sub-tree rooted at the D^i cluster. The nodes at the leaf level are called leaf clusters, and those at the non-leaf levels are called non-leaf clusters. The partition is recursively done until the unknown number in each cluster is no greater than $leafsize$, a predetermined constant.

4.2 Build elimination tree $\mathbf{E}_{\mathcal{I}}$ from $\mathbf{T}_{\mathcal{I}}$

The elimination tree of a matrix [2] is defined as the structure with n nodes $\{N^{(1)}, \dots, N^{(n)}\}$ such that node $N^{(p)}$ is the parent of $N^{(j)}$ if and only if the following is satisfied,

$$p = \min\{i > j | \mathbf{L}_{N^{(i)} \times N^{(j)}} \neq 0\} \tag{9}$$

in which $\mathbf{L}_{N^{(i)} \times N^{(j)}}$ is one matrix block in factor $\mathbf{L}$ with node $N^{(i)}$ being the row and node $N^{(j)}$ being the column. Because the sparse matrix $\mathbf{Y}$ obtained from a finite-element method is irreducible, the data structure generated from (9) is a tree.

In the proposed method, a domain cluster D^i in the cluster tree is decomposed and ordered by nested dissection as $\{D_1^{i+1}, D_2^{i+1}, S^{i+1}\}$. If we treat each cluster as a node in the elimination tree, it is easy to verify that node S^{i+1} is the parent of node D_1^{i+1} and D_2^{i+1} based on (9). Hence, the elimination tree generated from a nested-dissection based cluster tree is a tree with the interface cluster being the parent of subdomain clusters. Such a rule can be applied to each domain cluster recursively. The final elimination tree constructed for the entire unknown set has nonleaf nodes being the interface clusters at different levels and leaf nodes being the domain clusters of leaf size. As an example, the elimination tree corresponding to the cluster tree shown in Figure 2 is given in Figure 3. Note that each node $N^{(j)}$ in elimination tree has its own cluster tree structure. The elimination tree $E_{\mathcal{I}}$ is built upon selected nodes in $T_{\mathcal{I}}$. Combining all nodes in $E_{\mathcal{I}}$ will result in $\mathcal{I}$, the entire unknown set.

In an LU factorization procedure, from the definition of an elimination tree we can find that factorizing one node $N^{(j)}$ in the elimination tree, i.e. $\mathbf{Y}_{N^{(j)} \times N^{(j)}}$, will modify the matrix content belonging to all its ancestors. In other words,

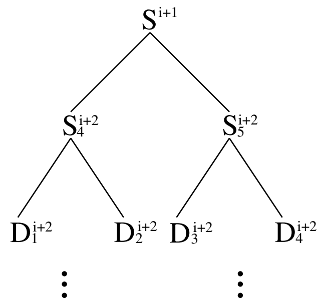

Figure 3: Illustration of an elimination tree.

$\mathbf{U}_{N^{(j)}\times ans(N^{(j)})}$ and $\mathbf{L}_{ans(N^{(j)})\times N^{(j)}}$ will be updated if we use $ans(N^{(j)})$ to denote the combined set of all ancestors of node $N^{(j)}$. The LU factorization of matrix $\mathbf{Y}$ is a bottom-up or a post-order traversal of the elimination tree $E_{\mathcal{I}}$.

4.3 Symbolic factorization

As mentioned in previous section, factorizing one node in elimination tree would only affect its ancestors. However, the combined set of all its ancestors is larger than what the factorization of one node actually modifies. We therefore need to find out the minimal set of nodes affected by the factorization of one node to save time and storage. In this paper, this minimal set is termed the boundary of each node, denoted by $bnd(N^{(j)})$ for each node $N^{(j)}$ in the elimination tree. As a simple example, a 3-level nested dissection on a domain cluster D^i is shown in Figure 4, where the interface cluster S^{i+1} at level $i+1$ is decomposed as $\{S_1^{i+1}, S_2^{i+1}, S_3^{i+1}\}$, and the S^{i+2} at level $i+2$ is decomposed as $\{S_1^{i+2}, S_2^{i+2}, S_3^{i+2}\}$. The boundary of D^{i+3}, denoted by $bnd(D^{i+3})$, can be identified geometrically as

$$bnd(D^{i+3}) = \{S_1^{i+1}, S_2^{i+1}, S_2^{i+2}, S_3^{i+2}, S^{i+3}\}. \tag{10}$$

Mathematically, the $\mathbf{L}_{D^{i+3}\times bnd(D^{i+3})}$ is filled due to the assembly of an FEM matrix or the factorization of lower-level nodes because the $bnd(D^{i+3})$ is in contact with node D^{i+3}, while we have

$$\mathbf{L}_{D^{i+3}\times\{ans(D^{i+3})-bnd(D^{i+3})\}} = 0. \tag{11}$$

Hence, if the above block is removed from memory as well as the factorization process, then the time and storage can be significantly reduced.

Identifying $bnd(N^{(j)})$ for each node $N^{(j)}$ in elimination tree based on geometrical information is straightforward. However, due to the complication of mesh and the irregularity in clusters having a small size, a geometrical way may not generate $bnd(N^{(j)})$ precisely. We hence develop the following symbolic factorization procedure to identify the boundary of each node in the elimination tree. Notice that a post-ordering or bottom-up tree traversal scheme is adopted to go through all nodes in the elimination tree.

```
Initialize $bnd(N^{(i)})$ from $\mathbf{Y}_{\mathcal{I}\times\mathcal{I}}$
For each node $N^{(i)}$ in elimination tree $E_{\mathcal{I}}$
    For each cluster $C_j$ (cluster index) in $bnd(N^{(i)})$
        $idx$ = node index of cluster $C_j$ in $E_{\mathcal{I}}$
        For each cluster $C_k \neq C_j$ in $bnd(N^{(i)})$
            Put $C_k$ into $bnd(N^{(idx)})$
        end
    end
end                                                        (12)
```

4.4 Generate the $\mathcal{H}$-matrix representation for each node in $E_{\mathcal{I}}$

After obtaining the boundary for each node in the elimination tree, the structure of the factorized matrix is known. This permits the construction of an $\mathcal{H}$-matrix representation $\mathbf{F}_j^{\mathcal{H}}$ for each node as the following

$$\mathbf{F}_j^{\mathcal{H}} = \begin{pmatrix} \mathbf{Y}^{\mathcal{H}}_{N^{(j)}\times N^{(j)}} & \mathbf{Y}^{\mathcal{H}}_{N^{(j)}\times bnd(N^{(j)})} \\ \mathbf{Y}^{\mathcal{H}}_{bnd(N^{(j)})\times N^{(j)}} & \hat{\mathbf{Y}}^{\mathcal{H}}_{bnd(N^{(j)})\times bnd(N^{(j)})} \end{pmatrix} \tag{13}$$

where $\mathbf{F}_j^{\mathcal{H}}$ is the $\mathcal{H}$-matrix representation of frontal matrix, the $\mathbf{Y}^{\mathcal{H}}_{N^{(j)}\times N^{(j)}}$, $\mathbf{Y}^{\mathcal{H}}_{N^{(j)}\times bnd(N^{(j)})}$, and $\mathbf{Y}^{\mathcal{H}}_{bnd(N^{(j)})\times N^{(j)}}$ are unique matrix blocks associated with node $N^{(j)}$. The $\hat{\mathbf{Y}}^{\mathcal{H}}_{bnd(N^{(j)})\times bnd(N^{(j)})}$ is composed of multiple pointers that link different parts to corresponding parts in the $\mathcal{H}$-matrix representation of the nodes that $bnd(N^{(j)})$ belongs to. Therefore, we build the $\mathcal{H}$-matrix representation for three matrix blocks and form the 4th one with the link to other matrices. The $\mathcal{H}$-matrix representation of $\mathbf{F}_j^{\mathcal{H}}$ is constructed based on the tree described in Section 4.1. Usually, the size of $bnd(N^{(j)})$ is larger than that of $N^{(j)}$, therefore, an adaptive division scheme is introduced to prevent skewed $\mathcal{H}$-matrix representations, namely a 2×1 or 1×2 division will be adopted if the size of row cluster and the size of column cluster are very different so that the row and column dimension can be made similar in an admissible block.

4.5 Numerical factorization and Solution

Numerical factorization is done as the following:

```
For each node $N^{(j)}$ in $E_{\mathcal{I}}$
    collect $\mathbf{F}_j^{\mathcal{H}}$, make it ready for factorization
    compute $\mathbf{L}^{\mathcal{H}}_{N^{(j)}\times N^{(j)}}$, $\mathbf{U}^{\mathcal{H}}_{N^{(j)}\times N^{(j)}}$ by $\mathcal{H}$-LU
        $\mathbf{Y}^{\mathcal{H}}_{N^{(j)}\times N^{(j)}} = \mathbf{L}^{\mathcal{H}}_{N^{(j)}\times N^{(j)}} \cdot \mathbf{U}^{\mathcal{H}}_{N^{(j)}\times N^{(j)}}$
    compute $\mathbf{U}^{\mathcal{H}}_{N^{(j)}\times bnd(N^{(j)})}$ by solving
        $\mathbf{L}^{\mathcal{H}}_{N^{(j)}\times N^{(j)}} \cdot \mathbf{U}^{\mathcal{H}}_{N^{(j)}\times bnd(N^{(j)})} = \mathbf{Y}^{\mathcal{H}}_{N^{(j)}\times bnd(N^{(j)})}$
    compute $\mathbf{L}^{\mathcal{H}}_{bnd(N^{(j)})\times N^{(j)}}$ by solving
        $\mathbf{L}^{\mathcal{H}}_{bnd(N^{(j)})\times N^{(j)}} \cdot \mathbf{U}^{\mathcal{H}}_{N^{(j)}\times N^{(j)}} = \mathbf{Y}^{\mathcal{H}}_{bnd(N^{(j)})\times N^{(j)}}$
    update $\hat{\mathbf{Y}}^{\mathcal{H}}_{bnd(N^{(j)})\times bnd(N^{(j)})}$ by
        $\hat{\mathbf{Y}}^{\mathcal{H}}_{bnd(N^{(j)})\times bnd(N^{(j)})} = \hat{\mathbf{Y}}^{\mathcal{H}}_{bnd(N^{(j)})\times bnd(N^{(j)})}$
                $- \mathbf{L}^{\mathcal{H}}_{bnd(N^{(j)})\times N^{(j)}} \cdot \mathbf{U}^{\mathcal{H}}_{N^{(j)}\times bnd(N^{(j)})}$
end                                                        (14)
```

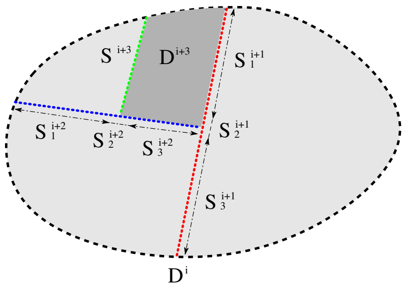

Figure 4: Illustration of symbolic factorization on multi-level domain cluster D^i

where the first three operations can be done based on $\mathcal{H}$-based arithmetic [4], and the last one is performed by developing an efficient algorithm to update the local $\mathcal{H}$-matrices associated with the nodes affected the node being factorized. With $\mathbf{L}/\mathbf{U}$ known in their $\mathcal{H}$-format, the solution can also be efficiently obtained for any right hand side.

5. COMPLEXITY AND ACCURACY

Consider a computational domain with the number of unknowns along each dimension being n. It is split into 8 smaller domains using nested dissection recursively. One intermediate stage is shown in Figure 5, where three shaded surface separators are combined to form one interface cluster S. The total number of unknowns is $N = n^3$, and the depth of elimination tree is $L = \log_2 n$. For a node $N^{(j)}$, the dimension of its frontal matrix $\mathbf{F}_j^{\mathcal{H}}$ satisfies

$$dim(\mathbf{F}_j^{\mathcal{H}}) = \#\{N^{(j)}\} + \#\{bnd(N^{(j)})\}. \quad (15)$$

As shown in Figure 5, the number of unknowns of the boundary for a cube domain with side length a is proportional to the surface area, $6a^2$, and the number of unknowns contained in the interface node is $3a^2$. Therefore, the following satisfies:

$$\#\{bnd(N^{(j)})\} = \alpha\#\{N^{(j)}\} \quad (16)$$

where α is a constant for an arbitraily-shaped domain, and equal to 2 for a cube domain. Assuming $\mathbf{F}_j$ is at level l with $l = L$ being the root level, its dimension can be approximated as $dim(\mathbf{F}_j) = 3\#\{N^{(j)}\} = 9 \times (2^l \times 2^l)$. It is proven that the complexity of an $\mathcal{H}$-matrix LU factorization is bounded by the complexity of an $\mathcal{H}$-matrix multiplication [4], which is $O(m \log^2 m)$ for a matrix of dimension m. For each node $N^{(j)}$ in the elimination tree $E_{\mathcal{I}}$, the operation associated with this node is hence

$$\begin{aligned} P_{lu}(\mathbf{F}_j^{\mathcal{H}}) &= P(\mathbf{F}_j^{\mathcal{H}} \otimes \mathbf{F}_j^{\mathcal{H}}) \\ &= O\big(2^l \times 2^l \log^2(2^l \times 2^l)\big) = O\big(2^{2l}(2l)^2\big). \end{aligned} \quad (17)$$

The total operation complexity is the summation of the operation complexity of each node in the elimination tree, thus

$$\begin{aligned} P_{lu}(\mathbf{Y}^{\mathcal{H}}) &= \sum_{j, N^{(j)} \in E_{\mathcal{I}}} P_{lu}(\mathbf{F}_j^{\mathcal{H}}) \\ &\sim \sum_{l=0}^{L} 8^{(L-l)} \times (2^l)^2 (2l)^2 = N \sum_{l=0}^{L} (\frac{4l^2}{2^l}) = O(N), \end{aligned} \quad (18)$$

which is linear. Similarly, the total storage complexity of the proposed solver can be evaluated as

$$\begin{aligned} Storage_{lu}(\mathbf{Y}^{\mathcal{H}}) &\sim \sum_{l=0}^{L} 8^{(L-l)} \times (2^l)^2 (2l) \\ &= N \sum_{l=0}^{L} (\frac{2l}{2^l}) = O(N). \end{aligned} \quad (19)$$

In the proposed solver, we represent the update and frontal matrices associated with each node in the elimination tree by an $\mathcal{H}$ matrix. Such an $\mathcal{H}$-matrix representation exists because the update and frontal matrices are related to the Schur complement. Since the original FEM matrix has an exact $\mathcal{H}$-matrix representation and its inverse has an error bounded $\mathcal{H}$-matrix representation [3], it can be proven that the intermediate Schur complements obtained during the factorization process and their inverses both can be represented by $\mathcal{H}$-matrices with error controlled.

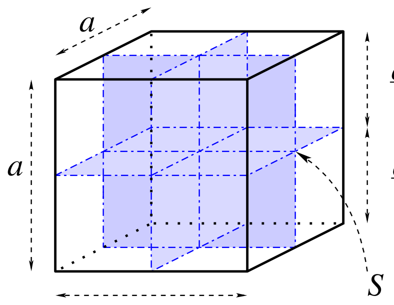

Figure 5: One step in nested dissection

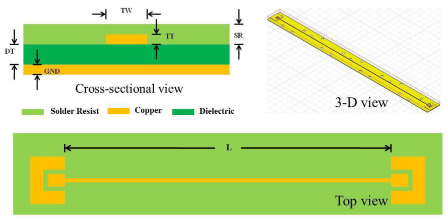

Figure 6: Illustration of a package interconnect.

6. NUMERICAL RESULTS

6.1 Package interconnect with measured data

We first validated the accuracy of the proposed method on a package interconnect provided by a company. Figure 6 illustrates its cross sectional view, top review, and 3-D view. Due to a nondisclosure agreement, detailed geometrical data are not shown. In Figure 7, we plot the S-parameters extracted by the proposed method in comparison with the measured data and the results generated from a commercial FEM-based tool. Excellent accuracy of the proposed solver can be observed in the entire frequency band.

6.2 Large-scale inductor array

We then simulated a large-scale 3-D inductor array [3] to examine the performance of the proposed direct solver. The computer used is a unix server with an AMD CPU running at 2.8 GHz. The frequency simulated is 10 GHz. The num-

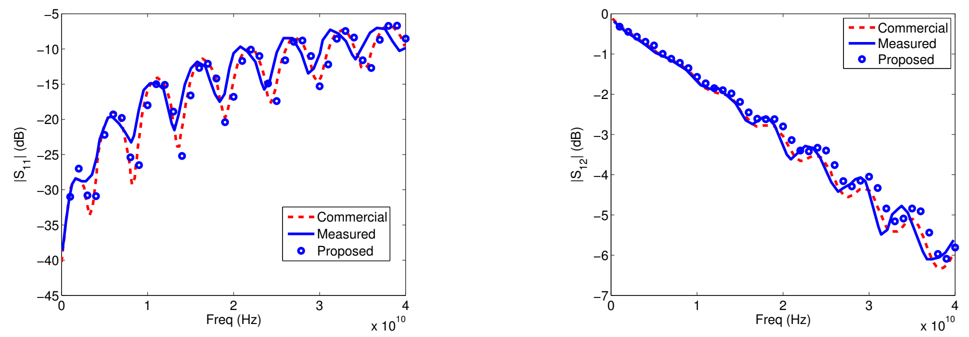

Figure 7: Simulation of a package interconnect. (a)$|S_{11}|$. (b)$|S_{12}|$.

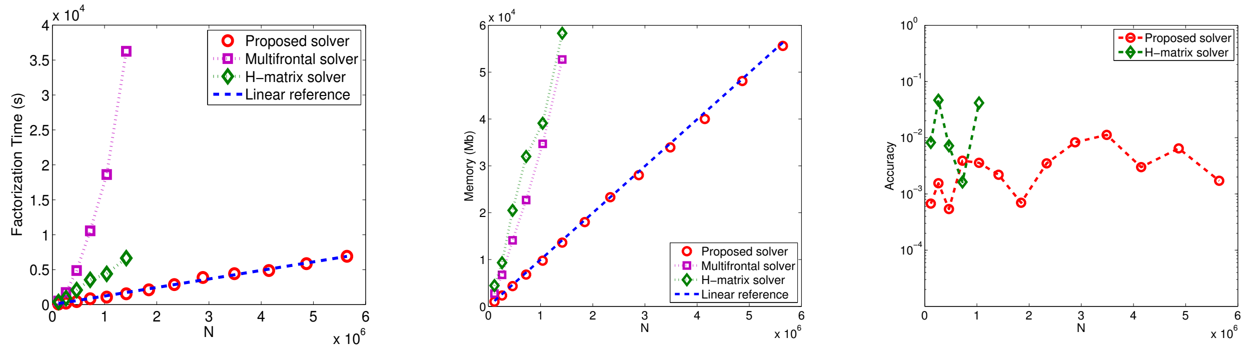

Figure 8: Simulation of a large-scale 3-D inductor array from 117,287 to 5,643,240 unknowns at 10 GHz. (a) LU factorization time. (b) Memory. (c) Accuracy.

ber of inductors varies from 2×2 to 14×14. The number of unknowns ranges from 117,287 to 5,643,240. The simulation parameters used are $leafsize$=8 and η=3. The relative error for (7) is set as $6e-5$. In Figure 8(a), (b), and (c), we plot the factorization time, the memory cost, and the accuracy of the proposed solver respectively with respect to N in comparison with both a state-of-the-art multifrontal based solver [5] and an $\mathcal{H}$-matrix based direct FEM solver. The accuracy is measured by relative residual. As can be seen from Figure 8, first, the proposed direct solver demonstrates a clear linear complexity with good accuracy achieved in the entire unknown range; second, the proposed method outperforms both the multifrontal based solver and the $\mathcal{H}$-matrix based solver. Notice that the multifrontal solver is based on exact arithmetic instead of prescribed accuracy, thus its accuracy is not shown in Fig. 8 (c). We also compared the performance of the proposed direct solver with a state-of-the-art iterative FEM solver available in the market. The computer used is a PC having an Intel CPU running at 2.4 GHz. Due to limited memory available on this PC, a 5×5 inductor array is simulated. To extract the S-parameters at 50 ports of the inductor array, the proposed direct solver only costs 558.39 s, whereas the commerical iterative FEM solver costs 2208 s.

7. CONCLUSION

A linear-complexity direct sparse matrix solution is developed for FEM-based large-scale 3-D circuit extraction with arbitrarily shaped lossy conductors in inhomoegenous materials. A comparison with both state-of-the-art direct FEM solvers and an iterative FEM solver has demonstrated its superior performance.

8. ACKNOWLEDGMENT

This work was supported by a grant from SRC (Task 1292.073). The authors would also like to thank Dr. Henning Braunisch at Intel for providing the interconnect structure and measured data.

9. REFERENCES

[1] A. George. Nested dissection of a regular finite element mesh. *SIAM J. on Numerical Analysis*, 10(2):345–363, April 1973.

[2] J. W. H. Liu. The multifrontal method for sparse matrix solution: Theory and practice. *SIAM Review*, 34(1):82–109, March 1992.

[3] H. Liu and D. Jiao. Existence of $\mathcal{H}$-matrix representations of the inverse finite-element matrix of electrodynamic problems and $\mathcal{H}$-based fast direct finite-element solvers. *IEEE Trans. MTT*, 58(12):3697–3709, December 2010.

[4] S. Borm, L. Grasedyck, and W. Hackbusch. *Hierarchical matrices*. Lecture note 21 of the Max Planck Institute for Mathematics, 2003.

[5] UMFPACK5.0. [on line] http://www.cise.ufl.edu/research/sparse/umfpack/.