Fast $\mathcal{H}$-Matrix-Based Direct Integral Equation Solver With Reduced Computational Cost for Large-Scale Interconnect Extraction

Wenwen Chai and Dan Jiao, *Senior Member, IEEE*

***Abstract*— In this paper, we propose a fast $\mathcal{H}$-matrix-based direct solution with a significantly reduced computational cost for an integral-equation-based capacitance extraction of large-scale 3-D interconnects in multiple dielectrics. We reduce the computational cost of an $\mathcal{H}$-matrix-based computation by simultaneously optimizing the $\mathcal{H}$-matrix partition to minimize the number of matrix blocks and minimizing the rank of each matrix block based on a prescribed accuracy. With the proposed cost-reduction method, we develop a fast LU-based direct solver. This solver possesses a complexity of $kC_{sp}O(N\log N)$ in storage, a complexity of $k^2C_{sp}^2O(N\log^2 N)$ in LU factorization, and a complexity of $kC_{sp}O(N\log N)$ in LU solution, where k is the maximal rank, C_{sp} is a constant dependent on matrix partition, and the constant kC_{sp} is minimized based on accuracy by the proposed cost-reduction method. The proposed solver successfully factorizes dense matrices that involve millions of unknowns in fast CPU time and modest memory consumption, and with the prescribed accuracy satisfied. As an algebraic method, the underlying fast technique is kernel independent.**

***Index Terms*— Capacitance extraction, direct solvers, fast integral equation solvers, $\mathcal{H}$ matrix, interconnect extraction, multiple dielectrics.**

I. Introduction

THE high level of integration has made the analysis and design of integrated circuits and packages increasingly challenging. In view of the increased design challenge, there exists an urgent need to reduce the computational complexity of existing methods for circuit extraction. Compared to a partial differential equation-based solver, a surface integral equation (IE) based solver reduces the number of unknowns and also naturally incorporates a radiation boundary condition. However, the IE-based analysis of 3-D interconnects generally leads to a dense system of linear equations. When a traditional direct method is used, the operation count is proportional to $O(N^3)$ and the memory requirement is proportional to $O(N^2)$, with N being the matrix size. When an iterative solver is used, the memory requirement remains the same, and the computing time is proportional to $O(N_{\rm it}N_{\rm rhs}N^2)$, where $N_{\rm it}$ denotes the total number of iterations required to reach convergence, and $N_{\rm rhs}$ is the number of right-hand sides. $N_{\rm it}$ is, in general, problem-, solver-, and accuracy-dependent. In state-of-the-art IE-based iterative solvers [1]–[7] for capacitance extraction, fast multipole method and hierarchical algorithms [1]–[3], [6], [7] were developed to perform a dense matrix-vector multiplication in $O(N)$ complexity, thereby significantly reducing the complexity of iterative solvers from $O(N_{\rm it}N_{\rm rhs}N^2)$ to $O(N_{\rm it}N_{\rm rhs}N)$. However, when $N_{\rm rhs}$ or $N_{\rm it}$ is large, iterative solvers become inefficient.

In [8]–[10], an $\mathcal{H}^2$-matrix-based mathematical framework was introduced to reduce the computational complexity of direct matrix solutions for the IE-based analysis of large-scale 3-D interconnects. The $\mathcal{H}^2$-matrix framework enables a highly compact representation and efficient computation of dense matrices [11]–[13]. The linear complexity $\mathcal{H}^2$-based dense matrix inversion was first established in [8] and [9]. In [10], it was also shown that an $\mathcal{H}^2$-based LU factorization can be performed in linear complexity. The resultant $O(N)$ direct IE solvers have demonstrated a clear advantage over state-of-the-art iterative solvers in both CPU time and memory consumption.

The storage complexity of an $\mathcal{H}^2$ matrix is $k^2C_{\rm sp}O(N)$, and the time complexity of an $\mathcal{H}^2$-based direct inverse is $k^3C_{\rm sp}^2O(N)$ [8], [9], where k denotes the maximum rank of admissible blocks, and $C_{\rm sp}$ is the maximum number of blocks that can be formed by a cluster in a block cluster tree, both of which are constant irrespective of N in a frequency-independent problem. If the rank of each admissible block can be minimized and the matrix partition can be optimized based on a prescribed accuracy, the constant k and $C_{\rm sp}$ in the complexity bound can be reduced. This will lead to a further acceleration of existing fast direct IE solvers. The $\mathcal{H}^2$-representation of an IE-based system matrix in [8]–[10] is generated by an interpolation-based method. The rank of each admissible block is determined by the number of interpolation points. Due to the limitation of an interpolation-based method, the resultant rank of each admissible block is often much larger than the minimal rank required to satisfy the prescribed accuracy. Furthermore, an interpolation-based scheme is not as flexible as a purely algebraic approach, since an efficient interpolation needs to take both the dimension and the geometry of a given problem into consideration. In addition, the matrix partition generated in [8]–[10] is a purely geometry-based one, which is not optimized based on accuracy.

Manuscript received March 6, 2012; revised September 22, 2012; accepted November 2, 2012. Date of publication January 9, 2013; date of current version January 31, 2013. This work was supported in part by a grant from SRC (Task 1292.073), and grants from NSF under Award 0747578 and Award 0702567. Recommended for publication by Associate Editor E.-P. Li upon evaluation of reviewers' comments.

The authors are with the School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47907 USA (e-mail: wchai@purdue.edu; djiao@purdue.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCPMT.2012.2228003

The major contribution of this paper is a new $\mathcal{H}$-matrix-based direct IE solver with the matrix partition optimized and the rank minimized based on a prescribed accuracy for the capacitance extraction of large-scale 3-D interconnects in multiple dielectrics. The $\mathcal{H}^2$ matrix is a special class of the $\mathcal{H}$ matrix [14]–[18]. In the proposed solver, by a theoretical analysis, we show the cost of the $\mathcal{H}$-matrix-based computation of an IE-based dense system is determined by both matrix partition and matrix block rank. We then develop a method to reduce the computational cost of the $\mathcal{H}$-based computation. This method is composed of three algorithms, each of which is performed based on a prescribed accuracy. The first algorithm is to minimize the rank of each admissible block; the second is to determine the minimal rank of each off-diagonal inadmissible block; and the third is to optimize the $\mathcal{H}$-partition to minimize the number of matrix blocks. The proposed algorithms are purely algebraic and have a linear computational cost for each matrix block. Based on the proposed cost reduction method, we develop an efficient LU-factorization for directly solving the dense system matrix resulting from an IE-based analysis of large-scale 3-D interconnects, with the constant $kC_{\rm sp}$ in the computational cost minimized based on accuracy. Numerical experiments have demonstrated the superior performance of the proposed direct IE solver.

II. Preliminaries

The $\mathcal{H}$-matrix-based methods are algebraic methods that are kernel independent. In the following, we use an integral equation for capacitance extraction in multiple dielectrics as an example to introduce the background of the $\mathcal{H}$-matrix-based methods.

A. Integral Equation for Capacitance Extraction in Multiple Dielectrics

Consider a multiconductor structure embedded in an inhomogeneous material. An IE-based solution for capacitance extraction results in the following dense system of equations [3], [8], [9]:

$$\mathbf{G}q = v \tag{1}$$

where $\mathbf{G} = \begin{bmatrix} \mathbf{P}_{\rm cc} & \mathbf{P}_{\rm cd} \\ \mathbf{E}_{\rm dc} & \mathbf{E}_{\rm dd} \end{bmatrix}$, $q = \begin{bmatrix} q_c \\ q_d \end{bmatrix}$, and $v = \begin{bmatrix} v_c \\ 0 \end{bmatrix}$, in which q_c and q_d are the charge vectors of the conductor panels and the dielectric–dielectric interface panels, respectively, and v_c is the potential vector associated with the conductor panels. The entries of $\mathbf{P}$ and $\mathbf{E}$ are

$$\begin{aligned} \mathbf{P}_{ij} &= \frac{1}{a_i}\frac{1}{a_j}\int_{S_i}\int_{S_j} g(r_i, r_j)dr_i dr_j \\ \mathbf{E}_{ij} &= (\varepsilon_a - \varepsilon_b)\frac{\partial}{\partial n_a}\frac{1}{a_i}\frac{1}{a_j}\int_{S_i}\int_{S_j} g(r_i, r_j)dr_i dr_j \end{aligned} \tag{2}$$

where a_i and a_j are the areas of panel S_i and S_j, g is the static Green's function, ε_a and ε_b are the permittivity of two adjacent regions a and b, and n_a is normal to the dielectric interface pointing to dielectric a. The diagonal entries of $\mathbf{E}_{\rm dd}$ are $e_{ij} = (\varepsilon_a + \varepsilon_b)/(2a_i\varepsilon_0)$.

In a uniform dielectric, (1) is reduced to

$$\mathbf{P}_{\rm cc}q_c = v_c. \tag{3}$$

B. $\mathcal{H}$-Matrix-Based Representation

The $\mathcal{H}$ (hierarchical) matrix is a general mathematical framework [14]–[18] which enables a highly compact representation and efficient numerical computation of dense matrices. Storage requirements and matrix–vector multiplications using $\mathcal{H}$-matrices for frequency-independent kernels have been shown to be of complexity $O(N\log N)$, and the matrix–matrix multiplications and matrix inversions using $\mathcal{H}$-matrices are of complexity $O(N\log^2 N)$ [14]. In the $\mathcal{H}$-matrix-based representation of a dense matrix, an admissibility condition [14] is used to partition the matrix blocks into admissible blocks that are low rank and inadmissible blocks that are full rank. Denoting the whole index set of the basis functions used for discretizing an IE-based equation by $\mathcal{I} := \{1, 2, \ldots, N\}$, consider two subsets t and s of $\mathcal{I}$, the admissibility condition is defined as [14, pp. 32–33]

$$\begin{aligned} &(t,s) \text{ are admissible} \\ &\text{if } \min\{\text{diam}(Q_t), \text{diam}(Q_s)\} \le \eta \ \text{dist}(Q_t, Q_s) \end{aligned} \tag{4}$$

where η is a positive parameter that can be used to control the admissibility condition, Q_t and Q_s are, respectively, the union of the supports of the basis functions residing in t and s, diam$(\cdot)$ is the Euclidean diameter of a set, and dist$(\cdot)$ is the Euclidean distance between two sets. Based on (4), a cluster tree and a block cluster tree [14] are constructed to efficiently carry out an $\mathcal{H}$-matrix partition.

Given a matrix $\mathbf{G}$, if all the blocks $\mathbf{G}^{t,s}$ formed by an admissible (t, s) in $\mathbf{G}$ can be represented by a factorized low-rank form

$$\mathbf{G}^{t,s}_{m,n} = \mathbf{A}_{m,k}\mathbf{B}^{\rm T}_{n,k}. \tag{5}$$

$\mathbf{G}$ has an $\mathcal{H}$-matrix representation. In (5), $k \in$ N is the rank of $\mathbf{G}^{t,s}$. For static problems, the rank k required by accuracy is bounded by a constant irrespective of matrix size. There are three representative methods for efficiently generating a rank-k representation of an IE-based dense matrix block: interpolation, Taylor expansion, and adaptive cross approximation (ACA) based scheme [14], [19].

III. Proposed Methods for Reducing the Computational Cost of an $\mathcal{H}$-Matrix-Based Solution of IE-Based Dense Matrices

A. Analysis on the Storage Requirement and Operation Counts of an $\mathcal{H}$-Based Solution of Dense Matrices

In an $\mathcal{H}$ matrix, each admissible block $\mathbf{G}_{m_i,n_i}$ has a factorized form $\mathbf{A}_{m_i\times k_i}\mathbf{B}^T_{n_i\times k_i}$ with rank $k_i < \min(m_i, n_i)$. The storage of each admissible block is thus reduced from $m_i \times n_i$ units to $k_i(m_i + n_i)$ units. By summing up the storage over all the admissible matrix blocks, we obtain

$$\text{storage} = \sum_{i=1}^{\rm nk} k_i(m_i + n_i) \tag{6}$$

where nk is the total number of admissible blocks. If the cluster tree used for the $\mathcal{H}$ matrix partition is a binary tree, (6) can be bounded as

$$\text{storage} \leq \sum_{l=0}^{P}\sum_{i=1}^{\text{nk}_l} k_l \left(\frac{N}{2^l}\times 2\right) = 2N\sum_{l=0}^{P}\frac{k_l nk_l}{2^l} \leq kC_{\text{sp}}O(N\log N) \quad (7)$$

where l is tree level, P is tree depth, $l = 0$ represents the root level, nk_l is the number of admissible blocks at level l, which is partition-dependent, and C_{sp} is the maximum number of blocks formed by one cluster in a block cluster tree. In (7), when deriving the first "$\leq$", we use the fact that the row/column dimension of a block at level l is $N/2^l$, and the rank of the admissible blocks in the same tree level l is bounded by k_l, while the rank bound at different tree levels can be different. In deriving the second "$\leq$" in (7), we utilize the following facts about k_l, nk_l, and P. First, for static problems, a constant rank k is sufficient to achieve the same order of accuracy irrespective of the size of the admissible block [14]. Second, nk_l is no greater than $2^l C_{\text{sp}}$. Third, the tree depth P is proportional to $\log_2 N$. Since the storage of the inadmissible blocks is $O(N)$ [14], the storage of an entire $\mathcal{H}$ matrix including both admissible and inadmissible blocks is still bounded by (7). The matrix–vector multiplication has the same complexity as storage.

The complexity of an $\mathcal{H}$-based matrix inversion is the same as that of an $\mathcal{H}$-based matrix–matrix multiplication [14]–[16]. In an $\mathcal{H}$-based matrix–matrix multiplication, the cost associated with each matrix block in the matrix product is $C_{\text{sp}}k_i^2(m_i + n_i)$ [14, pp. 127–130]. By summing up the cost of all the matrix blocks across all the tree levels, we obtain the operation counts for a matrix–matrix multiplication as the following:

$$\text{operation counts} = C_{\text{sp}}\sum_{l=0}^{P}\sum_{i=1}^{\text{nk}} k_i^2(m_i + n_i). \quad (8)$$

Similar to (7), we can obtain an asymptotic bound of (8) as

$$\text{operation counts} \leq C_{\text{sp}}\sum_{l=0}^{P}\sum_{l=0}^{P}\sum_{i=0}^{\text{nk}_l} k_i^2\left(\frac{N}{2^l}\times 2\right) = 2NC_{\text{sp}}\sum_{l=0}^{P}\sum_{l=0}^{P}\frac{k_l^2 nk_l}{2^l} \leq k^2C_{\text{sp}}^2 O(N\log^2 N). \quad (9)$$

From (7) and (9), it can be clearly seen that, to reduce the computational cost of an $\mathcal{H}$-based computation, we should reduce kC_{sp}. In existing $\mathcal{H}$-matrix-based solvers, as shown in Section II-B, the low-rank representation of each admissible block is generated by interpolation, Taylor expansion, or ACA-based approaches. The resultant rank is not the minimal rank required by accuracy. This is because, given an accuracy requirement, the rank obtained from singular value decomposition (SVD) is the minimum rank required by accuracy [20]. On the other hand, the $\mathcal{H}$-partition in existing $\mathcal{H}$-matrix-based solvers, and hence C_{sp}, is determined by a geometry-based admissibility condition shown in (4). This condition is controlled by an empirical parameter η, instead of a prescribed accuracy. The resultant kC_{sp} is not minimized for the prescribed accuracy. In [18], the $\mathcal{H}$-based block structure is improved by a coarsening procedure. However, the procedure only aims at reducing the storage of an $\mathcal{H}$-based matrix. In addition, the procedure has a large memory requirement for storing matrix blocks generated by ACA.

B. Proposed Algorithm for Reducing the Cost of an $\mathcal{H}$-Based Computation for IE-Based Capacitance Extraction

Based on the analysis above, in this section, from both rank and matrix partition perspectives, we propose a method to reduce the computational cost of an $\mathcal{H}$matrix-based method based on a prescribed accuracy for 3-D capacitance extraction in multiple dielectrics. This method simultaneously minimizes the number of admissible blocks and the rank of each admissible block. A pseudo-code of this method is shown in (10), which includes three essential algorithms: (ACA+)&RSVD, Rk-factor, and merge. The detail of each algorithm is given as follows:

Cost-Reduction of $\mathcal{H}$-based methods
Procedure **Cost_Mini**(b, ε) (the input block b is the entire $\mathcal{H}$-partition, ε denotes a prescribed accuracy)
If b is a non-leaf matrix block
for ($i = 0$;i <4;i++)
if $b(i)$ is an admissible block
(ACA+)&RSVD ($b(i)$, ε)
if $b(i)$ is an off-diagonal inadmissible block
Rk-Factor($b(i)$, ε)
if $b(i)$ is a non-leaf block
Cost_Mini($b(i)$, ε);
if all blocks in b are admissible blocks
Merge(b, ε) (10)

1) Algorithm 1: Reduced SVD performed on the factorized low-rank form obtained from ACA+ (ACA+&RSVD).

This algorithm is developed to efficiently minimize the rank of each admissible block based on a prescribed accuracy. If we directly apply SVD to the original full matrix to obtain its low-rank representation, although the resultant rank is minimal, the computational cost is high. Alternatively, we can use an interpolation, Taylor expansion, or ACA-based approach to efficiently convert a full-matrix block to a low-rank representation. However, the resultant rank is, in general, not the minimal one required by accuracy. In this paper, based on [14] and [18], we develop an algorithm to efficiently determine the minimal rank of each admissible block for a prescribed accuracy.

First, we use ACA+ to numerically obtain a factorized low-rank form of an admissible block. The ACA+ involves less storage and computational cost than ACA. The detailed procedure of ACA+ is very similar to that of the conventional ACA. The difference between them is as follows. At the beginning of an ACA+ algorithm, a reference row and a reference column of the original matrix are chosen to determine where to start the pivot search. A row and column pivot index is then determined from the reference ones. In the subsequent steps, the reference row and column can still be used. But if they are chosen as a pivot index, a new reference row and a new reference column must be chosen. This method only requires assembling k rows and k columns of an admissible block, where k is the rank determined by a certain accuracy requirement. The output of an ACA+ algorithm is $\mathbf{G}_{\text{mn}} = \mathbf{A}_{m,k}\mathbf{B}_{\text{nk}}^T$ where k is, in general, much less than m and n. The ACA+ algorithm terminates when

$$\left\|\mathbf{G} - \tilde{\mathbf{G}}\right\| = \left\|\mathbf{G} - \mathbf{A}\mathbf{B}^{\text{T}}\right\| \leq \varepsilon \left\|\mathbf{G}\right\| \tag{11}$$

is satisfied. Therefore, the error of the resultant $\mathcal{H}$-matrix representation is bounded by ε. After the ACA+ is completed, we obtain a factorized form $\mathbf{A}_{m,k}\mathbf{B}_{\text{nk}}^T$. For such a factorized form, SVD can be efficiently performed by a reduced SVD [14, pp. 108]. Hence, we apply reduced SVD to the factorized low-rank form to determine the actual rank that is needed to satisfy the accuracy requirement. By doing so, we keep the advantages of both SVD and ACA-based methods. The resultant rank is minimal, and, meanwhile, it is obtained in linear complexity for each admissible block.

2) Algorithm 2: Factorizing an off-diagonal inadmissible block to a low-rank form (Rk-factor).

Rk-factor is performed on an off-diagonal inadmissible block to minimize its rank for a given accuracy. It is possible that an off-diagonal block that is inadmissible in the matrix partition predetermined by (4) becomes admissible when its matrix information is considered. The function Rk-Factor is to factorize the full-matrix block to a rank-k matrix based on SVD and error tolerance ε.

3) Algorithm 3: Merge.

Whether the interaction between two geometrically separated blocks can be represented by a low-rank matrix or not is dependent on not only the geometry information, but also the matrix information. However, the admissibility condition given in (4) used for a traditional $\mathcal{H}$-partition is solely based on geometry without taking the matrix information into consideration. The resultant $\mathcal{H}$-partition is not optimal in terms of reducing the number of admissible blocks. Hence, we propose to merge multiple small admissible blocks to a single one based on a prescribed accuracy. By doing so, larger admissible blocks are generated at the parent levels of the small admissible blocks, thus reducing the total number of admissible blocks. To give an example, four admissible subblocks can be merged into one admissible block as follows:

$$\begin{aligned}
\begin{bmatrix} \mathbf{G}_1 & \mathbf{G}_2 \\ \mathbf{G}_3 & \mathbf{G}_4 \end{bmatrix} &= \begin{bmatrix} \mathbf{A}_1\mathbf{B}_1^{\text{T}} & \mathbf{A}_2\mathbf{B}_2^{\text{T}} \\ \mathbf{A}_3\mathbf{B}_3^{\text{T}} & \mathbf{A}_4\mathbf{B}_4^{\text{T}} \end{bmatrix} \\
&= \begin{bmatrix} \mathbf{A}_1 \\ 0 \end{bmatrix}\begin{bmatrix} \mathbf{B}_1 \\ 0 \end{bmatrix}^{\text{T}} + \begin{bmatrix} \mathbf{A}_2 \\ 0 \end{bmatrix}\begin{bmatrix} 0 \\ \mathbf{B}_2 \end{bmatrix}^{\text{T}} + \begin{bmatrix} 0 \\ \mathbf{A}_3 \end{bmatrix}\begin{bmatrix} \mathbf{B}_3 \\ 0 \end{bmatrix}^{\text{T}} + \begin{bmatrix} 0 \\ \mathbf{A}_4 \end{bmatrix}\begin{bmatrix} 0 \\ \mathbf{B}_4 \end{bmatrix}^{\text{T}} \\
&= \tilde{\mathbf{A}}_1\tilde{\mathbf{B}}_1^{\text{T}} + \tilde{\mathbf{A}}_2\tilde{\mathbf{B}}_2^{\text{T}} + \tilde{\mathbf{A}}_3\tilde{\mathbf{B}}_3^{\text{T}} + \tilde{\mathbf{A}}_4\tilde{\mathbf{B}}_4^{\text{T}} \\
&= \varepsilon\mathbf{A}\mathbf{B}^{\text{T}}
\end{aligned} \tag{12}$$

where the addition in the final step is carried out by the truncated addition operation in [14, p. 110], with the new rank k determined based on the accuracy ε. To determine whether to perform the merge operation shown in (12) or not, we compare the operation counts of the original children blocks with those of the new merged block. If the former is larger than the latter, we perform merging; otherwise, we do not perform merging, instead we keep the original children admissible blocks. To be more specific, we check whether $k^2(m+n) \leq \sum_{i=1}^{4} k_i^2(m_i+n_i)$ is satisfied or not, where k is the rank of the big block resulting from the merging operation, $m(n)$ is the row (column) dimension of the block, and k_i is the rank of each children admissible block. If the condition is satisfied, we merge blocks based on the prescribed accuracy; if not, we keep the original blocks. Therefore, by merging, the number of matrix blocks is reduced, as can be seen from (12). In addition, each merge operation is done to reduce $k_i^2(m_i+n_i)$, and hence the computational cost, as can be seen from (8). When preformed level by level, the entire computational cost of an $\mathcal{H}$-based direct solution is reduced, and thereby the kC_{sp} is reduced.

In (10), the (ACA+)&RSVD and Rk-factor minimize the rank k_i of each matrix block, and the merge operation minimizes the number of matrix blocks, with a prescribed accuracy satisfied. Each of the three algorithms reduces the $\mathcal{H}$-based storage and operations associated with one matrix block. Therefore, by traversing the entire matrix partition level by level, the entire storage and computational cost an $\mathcal{H}$-based solution is reduced. The proposed algorithms are purely algebraic, and hence are applicable to various formulations. In addition, the algebraic procedure has a linear-time cost for each block, and hence the computational overhead is small. A detailed cost analysis will be given in Section V.

It is worth mentioning that the ACA+ involved in the algorithm shown in (10) can be simply replaced by other rank-k generation methods, for example, the interpolation method. Such a method combined with reduced SVD can also efficiently reduce the rank of an admissible block.

IV. Proposed Fast Implementation of LU Factorization

In this section, we show how to perform a fast LU factorization using the $\mathcal{H}$-matrix-based representation of $\mathbf{G}$ and $\mathbf{G}$'s LU factors. The $\mathcal{H}$-based LU factorization has been discussed in [14, p. 119]. However, no detailed implementation is given. In the following, we give a number of pseudocodes to show a fast implementation of $\mathcal{H}$-based LU factorization. The fast $\mathcal{H}$-based LU factorization proposed in this paper has a factorization cost of $k^2C_{\text{sp}}^2O(N\log^2 N)$, a solution cost of $kC_{\text{sp}}O(N\log N)$, and memory consumption of $kC_{\text{sp}}O(N\log N)$, with constant kC_{sp} minimized for a prescribed accuracy by the method described in the section above.

A. LU Factorization Basics

Given an IE-based system matrix $\mathbf{G}$, we cast it into a form

$$\mathbf{G} = \begin{bmatrix} \mathbf{G}_{11} & \mathbf{G}_{12} \\ \mathbf{G}_{21} & \mathbf{G}_{22} \end{bmatrix}. \tag{13}$$

The LU decomposition can be recursively computed by the equation

$$\begin{aligned}
\mathbf{G} = \begin{bmatrix} \mathbf{G}_{11} & \mathbf{G}_{12} \\ \mathbf{G}_{21} & \mathbf{G}_{22} \end{bmatrix} &= \begin{bmatrix} \mathbf{L}_{11} & 0 \\ \mathbf{L}_{21} & \mathbf{L}_{22} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{U}_{11} & 0 \\ \mathbf{U}_{21} & \mathbf{U}_{22} \end{bmatrix} \\
&= \mathbf{L}\mathbf{U}.
\end{aligned} \tag{14}$$

B. Proposed Fast Implementation of the LU Factorization

We develop a pseudocode shown in (15) to recursively perform LU factorization

$$\left(\begin{array}{l}
\text{LU-Decomposition } \mathbf{G} = \mathbf{LU} \\
\text{Procedure } \textbf{H-LU(G)} \\
(\mathbf{G} \text{ is the input matrix overwritten by } \mathbf{L} \text{ and } \mathbf{U}) \\
\quad \textit{If } \mathbf{G} \text{ is a non-leaf block} \\
\qquad \textbf{H-LU}(\mathrm{G}_{11}) \rightarrow \mathbf{L}_{11}, \mathbf{U}_{11}, \\
\qquad \textbf{Solve-LX}(\mathrm{L}_{11}, \mathbf{G}_{12}) \rightarrow \mathbf{U}_{12}, \\
\qquad \textbf{Solve-XU}(\mathbf{G}_{21}, \mathbf{U}_{11}) \rightarrow \mathbf{L}_{21}, \\
\qquad -\mathbf{L}_{21} \times \mathbf{U}_{12} + \mathbf{G}_{22} \rightarrow \mathbf{G}_{22}, \\
\qquad \textbf{H-LU}(\mathbf{G}_{22}) \rightarrow \mathbf{L}_{22}, \mathbf{U}_{22}, \\
\quad \textit{else} \\
\qquad \textbf{Full-LU(G)}
\end{array}\right). \tag{15}$$

The underlying algorithm is as follows. When $\mathbf{G}$ is a non-leaf matrix block, we recursively call (15) until $\mathbf{G}_{11}$ is a full matrix block. We then directly compute the LU factors of the $\mathbf{G}_{11}$ using a full-matrix-based LU factorization, which generates $\mathbf{L}_{11}$ and $\mathbf{U}_{11}$. Next, we call function Solve-LX shown in (16) and Solve-XU to compute $\mathbf{U}_{12}$, and $\mathbf{L}_{21}$ respectively

$$\left(\begin{array}{l}
\textbf{Algorithm for Solving a Lower Triangular System} \\
\mathbf{LX} = \mathbf{G}, \text{ with } \mathbf{G} \text{ being an } \mathcal{H} \text{ matrix} \\
\text{Procedure } \textbf{Solve-LX(L,G)} \\
(\mathbf{L} \text{ and } \mathbf{G} \text{ are input matrices, } \mathbf{G} \text{ is overwritten by } \mathbf{X}) \\
\quad \textit{If } \mathbf{L} \text{ is a non-leaf block} \\
\qquad \textit{If } \mathbf{G} \text{ is a non-leaf block} \\
\qquad\quad \textbf{Solve-LX}(\mathbf{L}_{11}, \mathbf{G}_{11}), \textbf{Solve-LX}(\mathbf{L}_{11}, \mathbf{G}_{12}) \\
\qquad\quad -\mathbf{L}_{21} \times \mathbf{G}_{11} + \mathbf{G}_{21} \rightarrow \mathbf{G}_{21}, \textbf{Solve LX}(\mathbf{L}_{22}, \mathbf{G}_{21}) \\
\qquad\quad -\mathbf{L}_{21} \times \mathbf{G}_{12} + \mathbf{G}_{22} \rightarrow \mathbf{G}_{22}, \textbf{Solve LX}(\mathbf{L}_{22}, \mathbf{G}_{22}) \\
\qquad \textit{else if } \mathbf{G} \text{ is an admissible block} \\
\qquad\quad \textbf{Solve-LF(L,A)} \\
\qquad \textit{else} \\
\qquad\quad \textbf{Solve-LF(L,G)} \\
\quad \textit{else} \\
\qquad \textit{if } \mathbf{G} \text{ is an admissible block} \\
\qquad\quad \textbf{Solve-LF(L,A)} \\
\qquad \textit{else} \\
\qquad\quad \textbf{Full-LX}(\mathbf{L}, \mathbf{G}) \text{ (Solve a full-matrix} \\
\qquad\quad \text{triangular system)}
\end{array}\right). \tag{16}$$

The $\textbf{Solve-LX(L, G)}$ is to solve a lower triangular system $\mathbf{LX} = \mathbf{G}$, where $\mathbf{L}$ and $\mathbf{G}$ are input matrices having $\mathcal{H}$-representations, and $\mathbf{X}$ is the solution. The **Solve-XU(G, U)** is to solve an upper triangular system, which can be derived in a similar fashion as (16). In (16), a function **Solve-LF** is called. Similar to **Solve-LX**, **Solve-LF** also solves a triangular system. The difference is that the right-hand side matrix for **Solve-LX** is an $\mathcal{H}$ matrix, whereas that for **Solve-LF** is a full matrix. The pseudocode of **Solve-LF** is given in (17)

$$\left(\begin{array}{l}
\textbf{Algorithm for Solving a Lower Triangular System} \\
\mathbf{LX} = \mathbf{F}, \text{ with } \mathbf{F} \text{ being a Full Matrix} \\
\text{Procedure } \textbf{Solve-LF(L,F)} \\
(\mathbf{L} \text{ and } \mathbf{F} \text{ are input matrices, } \mathbf{F} \text{ is overwritten by } \mathbf{X}) \\
\quad \textit{If } \mathbf{L} \text{ is a non-leaf block} \\
\qquad \textbf{Solve-LF}(\mathbf{L}_{11}, \mathbf{F}_1) \\
\qquad -\mathbf{L}_{21} \times \mathbf{F}_1 + \mathbf{F}_2 \rightarrow \mathbf{F}_2 \\
\qquad \textbf{Solve-LF}(\mathbf{L}_{22}, \mathbf{F}_2) \\
\quad \textit{else} \\
\qquad \textbf{Full-LX}(\mathbf{L}, \mathbf{F}) \\
\qquad \text{(Solve a full-matrix triangular system)}
\end{array}\right). \tag{17}$$

In the final step of (15), we use $\mathbf{U}_{12}$ and $\mathbf{L}_{21}$ to update $\mathbf{G}_{22}$, and then call (15) recursively until $\mathbf{L}_{22}$ and $\mathbf{U}_{22}$ are computed. As can be seen from (13)–(17), efficient LU factorization relies on efficient block multiplications and block additions. In next subsections, we show how to efficiently perform these two operations for a prescribed accuracy.

C. Fast Implementation of the Block Multiplication $\boldsymbol{G}_b = \boldsymbol{G}_{b1} \times \boldsymbol{G}_{b2}$

We give a pseudocode of computing $\mathbf{G}_b = \mathbf{G}_{b1} \times \mathbf{G}_{b2}$ in (18)

$$\left(\begin{array}{l}
\textbf{Recursive Multiplication Algorithm} \\
\text{Procedure H-}\textbf{mult}(\mathbf{G}_{b1}, \mathbf{G}_{b2}, \mathbf{G}_b, \varepsilon_{LU}) \\
\quad \textit{If } \mathbf{G}_{b1}, \mathbf{G}_{b2}, \mathbf{G}_b \text{ are all non-leaf blocks} \\
\qquad \textit{for } (i = 0;\ i < 2;\ i{+}{+}) \\
\qquad\ \textit{for } (j = 0;\ j < 2;\ j{+}{+}) \\
\qquad\quad \textit{for } (k = 0;\ k < 2;\ k{+}{+}) \\
\qquad\qquad \text{H-}\textbf{mult}(\mathbf{G}_{b1}(i,k), \mathbf{G}_{b2}(k,j), \mathbf{G}_b(i,j), \varepsilon_{LU}) \\
\quad \textit{else if } \mathbf{G}_b \text{ is a non-leaf block, } \mathbf{G}_{b1} \text{ or } \mathbf{G}_{b2} \text{ is a} \\
\quad \text{leaf block} \\
\qquad \textbf{Multiply-RK}(\mathbf{G}_{b1}, \mathbf{G}_{b2}, \tilde{\mathbf{G}}_b, \varepsilon_{LU}) \\
\qquad \mathbf{G}_b = \tilde{\mathbf{G}}_b + \mathbf{G}_b \text{ (based on } \varepsilon_{LU}) \\
\quad \textit{else if } \mathbf{G}_b \text{ is an admissible block} \\
\qquad \textbf{Multiply-RK}(\mathbf{G}_{b1}, \mathbf{G}_{b2}, \mathbf{G}_b, \varepsilon_{LU}) \\
\quad \textit{else if } \mathbf{G}_b \text{ is an inadmissible block} \\
\qquad \textbf{Multiply-Full}(\mathbf{G}_{b1}, \mathbf{G}_{b2}, \mathbf{G}_b)
\end{array}\right) \tag{18}$$

where b, b_1, and b_2 represent three blocks in the same level of an $\mathcal{H}$-partition, and $\varepsilon_{\mathrm{LU}}$ represents a prescribed accuracy. If $\mathbf{G}_b$, $\mathbf{G}_{b1}$, and $\mathbf{G}_{b2}$ are all non-leaf blocks, we recursively call (18). If one of $\mathbf{G}_{b1}$ and $\mathbf{G}_{b2}$ is a leaf block, or $\mathbf{G}_b$ is an admissible block, we call function **Multiply-Rk** shown in (19) to compute an admissible product. In (18), the addition is performed based on the prescribed accuracy $\varepsilon_{\mathrm{LU}}$. The detailed procedure of the addition is given in the following subsection:

$$\left(\begin{array}{l}
\text{Procedure } \textbf{Multiply-RK}(\mathbf{G}_{b1}, \mathbf{G}_{b2}, \mathbf{G}_b, \varepsilon_{LU}) \\
\quad \textit{if } \mathbf{G}_{b1} \text{ and } \mathbf{G}_{b2} \text{ are both non-leaf blocks} \\
\qquad \textit{for } (\ i = 0;\ i < 2;\ i{+}{+}) \\
\qquad\ \textit{for } (\ j = 0;\ j < 2;\ j{+}{+}) \\
\qquad\quad \textit{for } (\ k = 0;\ k < 2;\ k{+}{+}) \\
\qquad\qquad \textbf{Multiply RK}(\mathbf{G}_{b1}(i,k), \mathbf{G}_{b2}(k,j), \\
\qquad\qquad \tilde{\mathbf{G}}_b(i,j), \varepsilon_{LU}) \\
\qquad\qquad \mathbf{G}_b = \tilde{\mathbf{G}}_b + \mathbf{G}_b \text{ (based on } \varepsilon_{LU}) \\
\qquad\qquad (\tilde{\mathbf{G}}_b \text{ is a non-leaf block}) \\
\quad \textit{else if } \mathbf{G}_{b1} \text{ or } \mathbf{G}_{b2} \text{ is an admissible block} \\
\qquad \mathbf{G}_{b1}\mathbf{AB}^T \rightarrow (\mathbf{G}_{b1}\mathbf{A})\mathbf{B}^T = \tilde{\mathbf{A}}_b\mathbf{B}^T = \tilde{\mathbf{G}}_b \\
\qquad (\tilde{G}_b \text{ is an admissible block}) \\
\qquad \mathbf{G}_b = \tilde{G}_b + \mathbf{G}_b \text{ (based on } \varepsilon_{LU}) \\
\quad \textit{else if } \mathbf{G}_{b1} \text{ or } \mathbf{G}_{b2} \text{ is an inadmissible block} \\
\qquad \mathbf{G}_{b1}\mathbf{G}_{b2} = \mathbf{G}_{b1}\mathbf{F} \rightarrow (\mathbf{G}_{b1}\mathbf{A})\mathbf{B}^T = \tilde{A}_b\mathbf{B}^T = \tilde{\mathbf{G}}_b \\
\qquad \text{(based on } \varepsilon_{LU}) \\
\qquad \mathbf{G}_b = \tilde{G}_b + \mathbf{G}_b \text{(based on } \varepsilon_{LU})
\end{array}\right). \tag{19}$$

In (20), there are two multiplication cases. One is to multiply an admissible block $\mathbf{G}_{b1}$ by an admissible block of an $\mathbf{AB}^T$ form, for which we can compute $\mathbf{G}_{b1}\mathbf{A}$ as a new $\mathbf{A}$. The other multiplication case is to multiply $\mathbf{G}_{b1}$ by a full matrix block $\mathbf{F}$, for which we can first apply SVD to $\mathbf{F}$ to generate a form $\mathbf{AB}^T$ based on the prescribed accuracy $\varepsilon_{\mathrm{LU}}$. If $\mathbf{G}_b$ is a full matrix block, a normal full matrix multiplication is computed. The additions in (19) again are performed based on $\varepsilon_{\mathrm{LU}}$.

D. Fast Implementation of the Block Addition $\boldsymbol{G}_b = \boldsymbol{G}_{b1} + \boldsymbol{G}_{b2}$

Two cases are involved in the addition operations.

Case 1: If $\mathbf{G}_b$, $\mathbf{G}_{b1}$, and $\mathbf{G}_{b2}$ have the same $\mathcal{H}$-partition, the addition can be done using the following procedure.

1) If three blocks are all full matrices, we simply add two full matrices up.
2) If three blocks are all admissible matrices, for example, $\mathbf{G}_{b1} = \mathbf{A}_{b1}\mathbf{B}_{b1}^T$ with rank k_1, $\mathbf{G}_{b2} = \mathbf{A}_{b2}\mathbf{B}_{b2}^T$ with rank k_2, and $\mathbf{G}_b = \mathbf{A}_b\mathbf{B}_b^T$, the $\mathbf{G}_b = \mathbf{G}_{b1} + \mathbf{G}_{b2}$ can be realized by a truncated addition operation using the

approach shown in [14, p. 110]. The rank k of the resultant $\mathbf{G}_b$ is adaptively determined by the prescribed accuracy $\varepsilon_{\mathrm{LU}}$.

3) If three blocks are all non-leaf blocks, the addition can be carried out by summing over all the inadmissible blocks using 1), and all the admissible ones using 2).

Case 2: If the three blocks do not share the same partition, we convert the $\mathcal{H}$-matrix partitions of $\mathbf{G}_{b1}$ and $\mathbf{G}_{b2}$ both into the partition of $\mathbf{G}_b$. Take the block $\mathbf{G}_{b1}$ as an example. If $\mathbf{G}_{b1}$ is an admissible block but $\mathbf{G}_b$ is a non-leaf block that has four admissible subblocks, we convert $\mathbf{G}_{b1}$ by the formula

$$\mathbf{G}_{b1} = \mathbf{A}_{b1}\mathbf{B}_{b1}^{\mathrm{T}} = \begin{bmatrix} \tilde{\mathbf{A}}_1 \\ \tilde{\mathbf{A}}_2 \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{B}}_1 \\ \tilde{\mathbf{B}}_2 \end{bmatrix}^{\mathrm{T}} \tag{20}$$

$$= \begin{bmatrix} \tilde{\mathbf{A}}_1\tilde{\mathbf{B}}_1^{\mathrm{T}} & \tilde{\mathbf{A}}_1\tilde{\mathbf{B}}_2^{\mathrm{T}} \\ \tilde{\mathbf{A}}_2\tilde{\mathbf{B}}_1^{\mathrm{T}} & \tilde{\mathbf{A}}_2\tilde{\mathbf{B}}_2^{\mathrm{T}} \end{bmatrix} = \tilde{\mathbf{G}}_{b1} \tag{21}$$

where $\tilde{\mathbf{G}}_{b1}$ contains four admissible subblocks, which is exactly equal to $\mathbf{G}_{b1}$. The opposite procedure, where $\mathbf{G}_b$ is an admissible block while $\mathbf{G}_{b1}$ contains four admissible subblocks, can be performed by the scheme shown in (11).

V. Total Computational Cost Analysis

Two numerical procedures are involved in the proposed direct IE solver: reduction of the computational cost by simultaneously optimizing the matrix partition, and minimizing the rank and LU-based direct matrix solution. In the proposed cost reduction method, as described in Section III-B, there are three algorithms. The first algorithm (ACA+ and reduced SVD) has a linear cost for each admissible block [14], [19]; the second algorithm has a constant cost for each block since SVD is used to do the factorization of off-diagonal inadmissible blocks, which have a constant size (*leafsize*). In the third algorithm, the conversion of non-leaf blocks to an admissible block shown in (11) is carried out by the function Merge using a reduced-SVD-based truncated addition, which has a linear cost for each matrix block. As a result, the total cost of the proposed cost reduction method is $O(N\log N)$, which is negligible compared to matrix factorization. For the LU-based direct solution, as can be seen from (15), at the leaf level, the computation of the recursive LU factorization essentially includes a full-matrix LU factorization, a full-matrix solution of a lower triangular system, and a full-matrix solution of an upper triangular system, all of which have the same computational cost as a full-matrix block multiplication. At all the other levels, a number of block–block multiplications are computed, which have the same recursive pattern as that in an $\mathcal{H}$-based matrix-matrix multiplication. Therefore, the $\mathcal{H}$-based LU factorization has the same cost as an $\mathcal{H}$-based multiplication, which is bounded by $k^2C_{\mathrm{sp}}^2O(N\log^2 N)$, where constant kC_{sp} is reduced in this paper based on a prescribed accuracy. The $\mathcal{H}$-based LU solution has the same complexity as an $\mathcal{H}$-based matrix-vector multiplication, which is $kC_{\mathrm{sp}}O(N\log N)$.

The storage and time complexity of an $\mathcal{H}^2$-based direct solver in [8] and [9] are $k^2C_{\mathrm{sp}}O(N)$ and $k^3C_{\mathrm{sp}}^2O(N)$, respectively. Although the complexity is linear, the constant k and C_{sp} are not minimized based on accuracy. Therefore, for a given accuracy, the cost of an $\mathcal{H}^2$-based direct solver with large k and C_{sp} can be larger than the cost of the proposed direct solver with k and C_{sp} minimized based on accuracy.

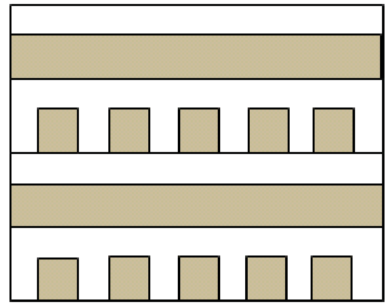

Fig. 1. Illustration of a four-layer 3-D bus structure.

VI. Numerical Results

A number of cases were simulated to validate the performance of the proposed cost reduction method and the resultant $\mathcal{H}$-based fast direct IE solver for large-scale interconnect extraction. For all these simulations, $\eta = 2$ and *leafsize* $= 20$ were used. The error tolerance ε used in (10) for optimizing the partition and minimizing the rank was set as 10^{-3}. The error tolerance $\varepsilon_{\mathrm{LU}}$ used in the LU factorization was 10^{-2}. The computer used was a Dell PowerEdge 6950s server with an 8222SE AMD Opteron processor running at 3 GHz.

A. Four-Layer 3-D Bus Structure in a Uniform Dielectric

Fig. 1 shows a four-layer 3-D bus structure. At each layer, there are p conductors, and each conductor has a dimension of $1 \times 1\times (2p+1)$ m^3, where p is chosen from 5, 10, 20 to 40. The resultant number of unknowns is from 3680 to 208 640. In Fig. 2(a), for the case of $p = 40$, we plot the maximal rank k among all admissible blocks at the lowest tree level where the admissible block size is the largest. Two methods are used to obtain the maximal rank k: the proposed scheme (ACA+ and SVD) and ACA+ only. An obvious rank reduction by using the proposed method can be seen from Fig. 2(a). The matrix accuracy is 7.57×10^{-4} without the proposed rank minimization, and 7.81×10^{-4} with the proposed minimization. The accuracy is measured by $||\mathbf{G}\text{-}\tilde{\mathrm{G}}||_F/||\mathbf{G}||_F$, where $\mathbf{G}$ is the original matrix, $\tilde{\mathrm{G}}$ is an $\mathcal{H}$-based representation, and the subscript F denotes a Frobenius norm. It is clear that the rank is reduced by the proposed method without sacrificing accuracy. We have also used the interpolation based scheme employed in [8]–[10] to obtain the rank of each admissible block. The resultant rank is higher than that generated by ACA+ for the same accuracy, and hence higher than the proposed method. In Fig. 2(b), we plot the maximum number of admissible blocks that can be formed by one cluster in a block cluster tree (C_{ad}) produced by the conventional $\mathcal{H}$ partition constructed based on (4), which is also the partition scheme used in [8]–[10], and the C_{ad} generated by the proposed optimized $\mathcal{H}$matrix partition. Clearly, the C_{ad} is reduced significantly. Since the proposed $\mathcal{H}$-partition optimization does not increase the number of inadmissible blocks, the C_{sp} which is the sum of C_{ad} and the number of inadmissible blocks, is also reduced significantly.

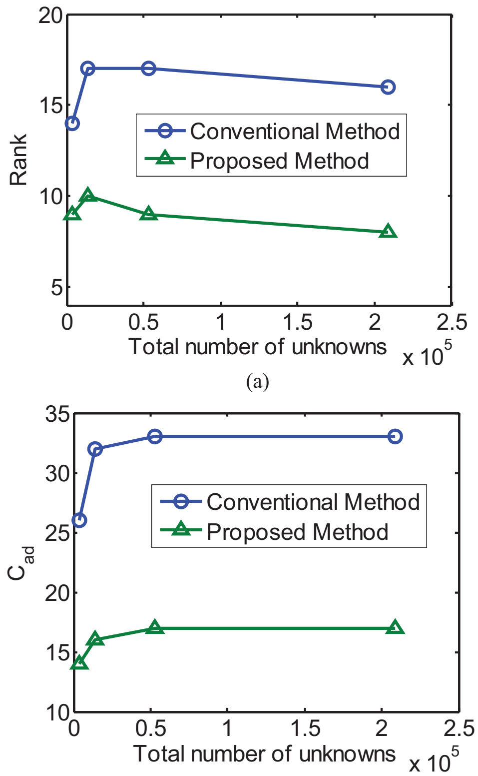

Fig. 2. Performance of the proposed method for partition optimization and rank minimization in analyzing a four-layer bus structure in a uniform material. (a) Maximal rank versus N. (b) C_{ad} versus N.

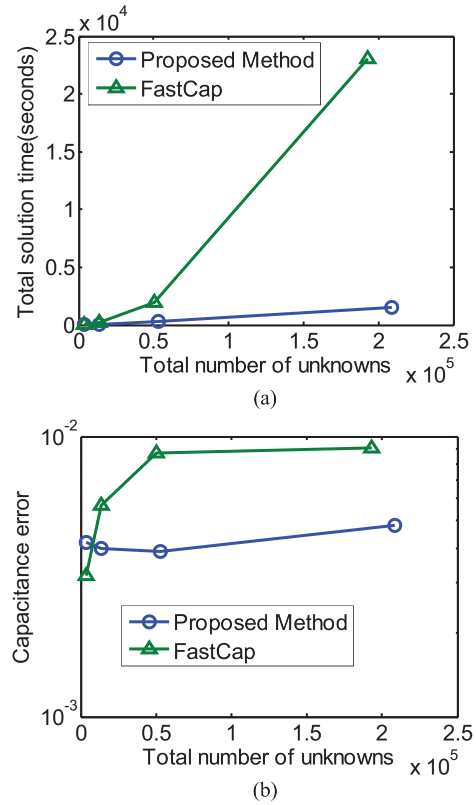

Fig. 3. Performance of the proposed LU-based direct IE solver for simulating a four-layer bus structure in a uniform material. (a) Total solution time. (b) Capacitance error.

From Fig. 2, it is evident that both k and C_{sp} are minimized to be a small number based on the prescribed accuracy. As a result, the cost of the $\mathcal{H}$-based computation is reduced in both storage and CPU time as can be seen from (7) and (9).

Next, we test the effectiveness of the proposed LU-based direct solver for simulating the $m \times m \times m \times m$ bus structure shown in Fig. 1. In Fig. 3(a), we plot the total solution time of the proposed LU-based direct solver, including the construction time of the proposed $\mathcal{H}$-matrix representation that has an optimized partition and minimized rank, LU decomposition time, and LU solution time. For comparison, we also plot the total solution time using FastCap2.0, which is available in the public domain. When using FastCap2.0, the expansion order is chosen as 2, the convergence tolerance is set to be 0.1%, and a similar number of unknowns are generated. As can be seen from Fig. 3(a), the proposed direct solver is faster than FastCap2.0. In addition, FastCap2.0 does not exhibit a linear scaling although it performs a dense matrix–vector multiplication in linear complexity. This is attributed to the increased number of iterations and increased number of right-hand sides when the number of unknowns increases. In Fig. 3(b), we plot the accuracy of the extracted capacitance matrix with respect to the number of unknowns. The capacitance accuracy is measured by $||\mathbf{C}\text{-}\mathbf{C}'||_F/||\mathbf{C}||_F$, where $\mathbf{C}$ is the capacitance matrix obtained from FastCap2.0 with a higher expansion order 3, and $\mathbf{C}'$ is that generated by the proposed solver or by FastCap2.0 with expansion order 2. As can be seen, the proposed solver reduces the total solution time without compromising in accuracy.

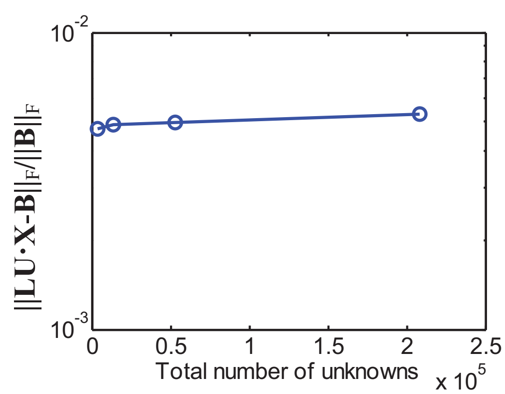

Fig. 4. LU solution accuracy for simulating a four-layer bus structure in a uniform material.

In Fig. 4, we plot the LU solution accuracy measured by $||\mathbf{LU}\cdot\mathbf{X}\text{-}\mathbf{B}||_F/||\mathbf{B}||_F$, where each column b_i of $\mathbf{B}$ represents one right-hand side, and each column x_i of $\mathbf{X}$ is the solution of (1) corresponding to b_i with $i = 1, 2, \ldots, N_{\mathrm{con}}$, where N_{con} is the total number of conductors. Excellent accuracy can be observed. In addition, the accuracy is kept to be almost a constant in the entire unknown range. Fig. 5 shows the memory of the proposed direct solver. An almost linear complexity can be observed.

B. Four-Layer 3-D Bus Structure in Multiple Dielectrics

The second example is a four-layer 3-D bus structure, similar to that shown in Fig. 1, but embedded in multiple

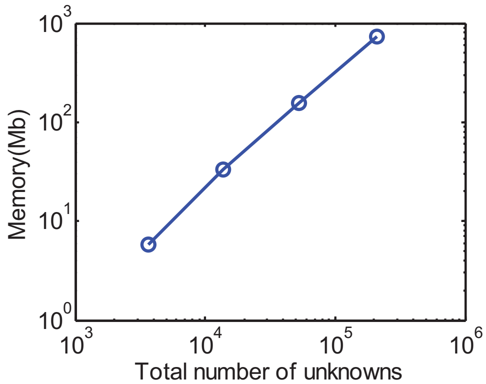

Fig. 5. Memory consumption for simulating a four-layer bus structure in a uniform material.

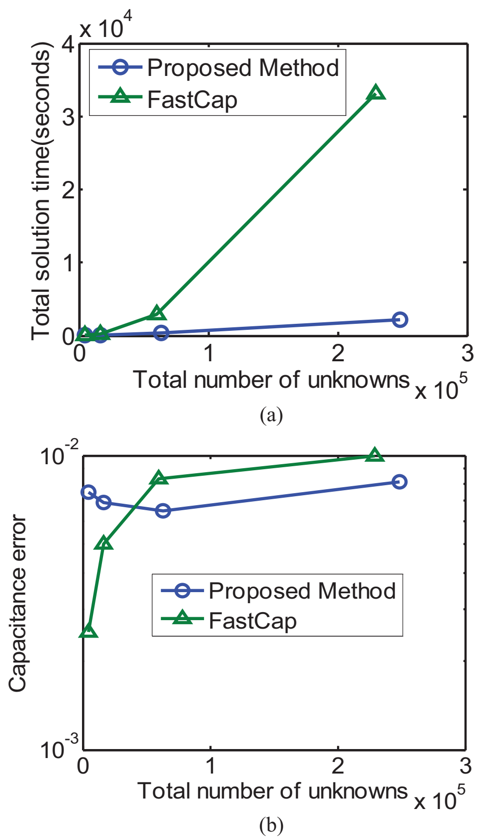

Fig. 6. Performance of the proposed LU-based direct solver for simulating a four-layer bus structure in multiple dielectrics. (a) Total solution time. (b) Capacitance error.

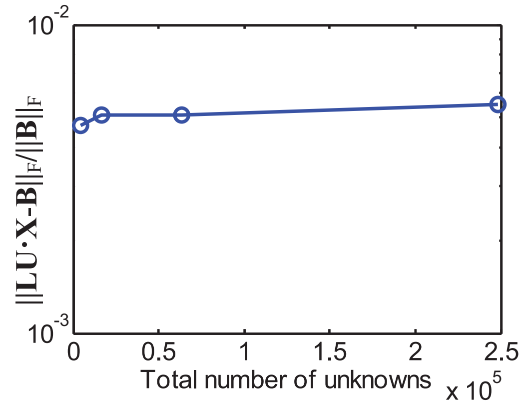

Fig. 7. LU solution accuracy for simulating a four-layer bus structure in multiple dielectrics.

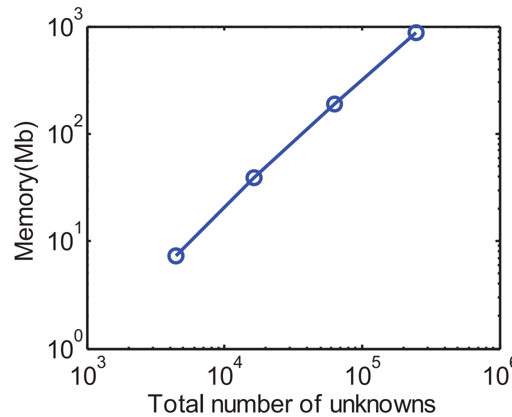

Fig. 8. Memory consumption for simulating a four-layer bus structure in multiple dielectrics.

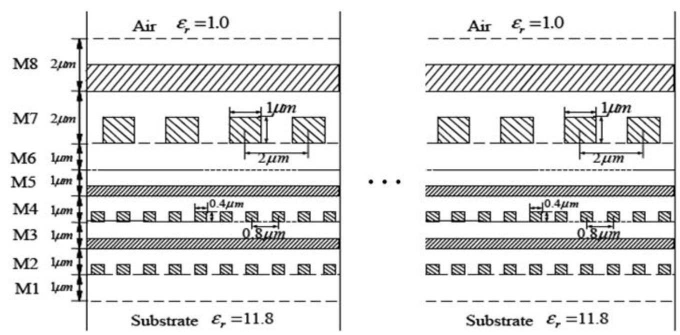

Fig. 9. 3-D large-scale M1–M8 on-chip interconnect.

dielectrics. The relative permittivity of each layer from bottom to top is 3.9, 2.5, 7.0, and 1.0, respectively. The conductor number in each layer, p, is chosen from 5, 10, 20 to 40. The resultant number of unknowns ranges from 4472 to 248 492. In Fig. 6(a), we plot the total solution time of the proposed direct solver. An almost linear complexity can be observed. For comparison, the total solution time of FastCap2.0 is also plotted. The advantage of the proposed solver can be clearly seen. The accuracy of the capacitance matrix extracted by both solvers is shown in Fig. 6(b). It is clear that the proposed method reduces the CPU time without sacrificing accuracy.

The accuracy of the proposed LU solution is shown in Fig. 7, from which an excellent accuracy can be seen. In Fig. 8, we plot the memory consumption of the proposed direct IE solver. Again, an almost linear complexity can be observed.

C. Large-Scale 3-D M1–M8 On-Chip Interconnects

To test the performance of the proposed direct solver in simulating very large cases, we simulate a multilayer 3-D on-chip interconnect structure [3] shown in Fig. 9. The relative permittivity is 3.9 in M1, 2.5 from M2 to M6, and 7.0 from M7 to M8. We simulate a suite of such structures, in which

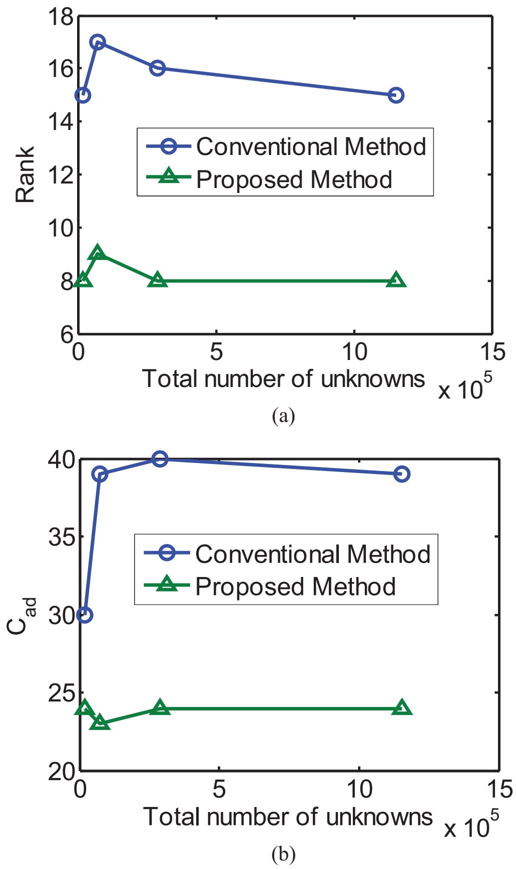

Fig. 10. Performance of the proposed method for partition optimization and rank minimization on a 3-D large-scale M1–M8 interconnect embedded in multiple dielectrics. (a) Maximal rank versus N. (b) C_{ad} versus N.

the number of wires is increased from 48, 96, 192, to 384 conductors. The largest structure has over 1 million unknowns.

In Fig. 10(a), we plot the maximal rank among all admissible blocks that are located at the lowest level with the proposed scheme (ACA+ and SVD) and with ACA+. The rank is greatly reduced by the proposed method. In Fig. 10(b), we plot C_{ad} in the original $\mathcal{H}$ partition and that in the proposed optimized $\mathcal{H}$-partition. Clearly, the C_{ad} is reduced significantly.

In Fig. 11(a) and (b), we plot the memory and the total solution time of the proposed direct solver. As can be seen, an almost linear scaling can be observed for both memory and CPU time of the proposed direct solver. The capacitance error shown in Fig. 11(c) is measured by $||\mathbf{C}–\mathbf{C}'||_F/||\mathbf{C}||_F$, where $\mathbf{C}'$ is obtained by the proposed solver with $\varepsilon = 10^{-3}$ and $\varepsilon_{\mathrm{LU}} = 10^{-2}$, and reference $\mathbf{C}$ is obtained with a higher order accuracy setting of $\varepsilon = 10^{-4}$ and $\varepsilon_{\mathrm{LU}} = 10^{-3}$ from the proposed solver. We were not able to use other capacitance solvers to generate reference $\mathbf{C}$ for such a large example. As can be seen from Fig. 11(c), excellent accuracy in capacitance is observed across the entire unknown range.

In Fig. 11, we also compare the performance of the $\mathcal{H}^2$-based direct solver in [8] and [9] with that of the proposed direct solver in memory, total solution time, and capacitance error. The $\mathcal{H}^2$-rerpesentation in [8] and [9] is generated by an interpolation-based method. Since the number of interpolation

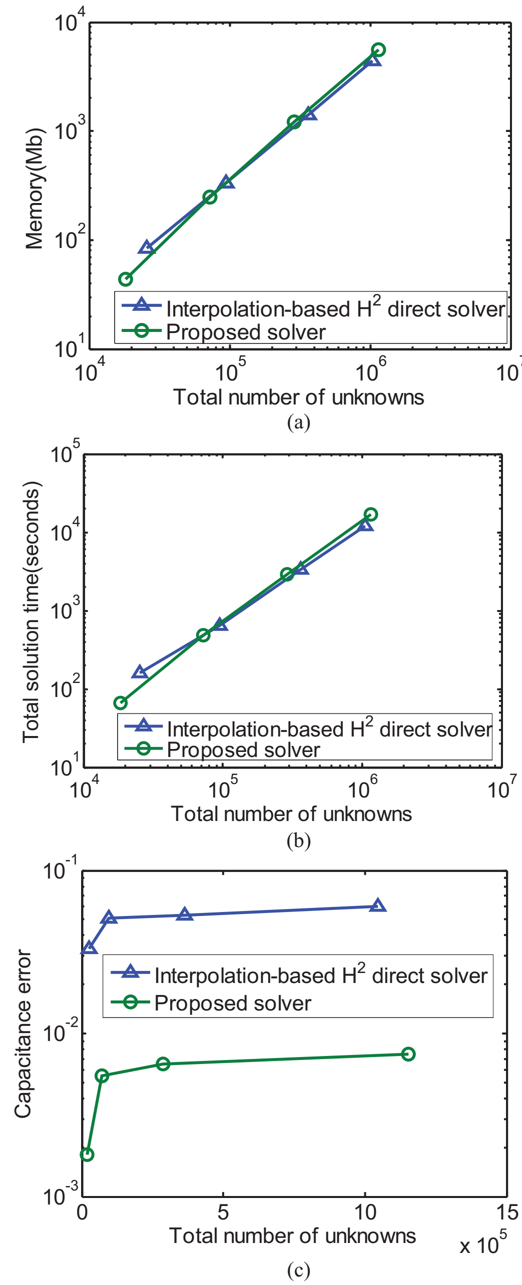

Fig. 11. Performance of the proposed LU-based direct IE solver for simulating a 3-D large-scale M1–M8 on-chip interconnect embedded in multiple dielectrics. (a) Memory. (b) Total solution time. (c) Capacitance error.

points used in [8] and [9] for this example, and thereby the rank of the $\mathcal{H}^2$-rerpesentation, is 1, the resulting accuracy is not as good as that of the proposed solver. From Fig. 11, it is clear that to achieve the same level of accuracy as the proposed solver, the $\mathcal{H}^2$-based direct solver in [8] and [9] would cost more in CPU time and memory since the number of interpolation points will have to be increased from 1 to at least 2. Therefore, the rank will become 4 times larger.

VII. Conclusion

This paper presented a fast $\mathcal{H}$-matrix-based direct solution of $kC_{\mathrm{sp}}O(N\log N)$ complexity in storage, $k^2C_{\mathrm{sp}}^2O(N\log^2N)$

complexity in LU factorization, and $kC_{\rm sp}O(N\log N)$ complexity in LU solution, with constant $kC_{\rm sp}$ minimized based on accuracy by developing an algorithm that simultaneously optimizes the $\mathcal{H}$-matrix partition and minimizes the rank of each matrix block. Applications to large-scale capacitance extraction in multiple dielectrics have demonstrated the effectiveness of the proposed algorithm in minimizing the number of matrix blocks and the rank of each matrix block. The proposed direct solver has also shown a clear advantage in computational efficiency over a state-of-the-art iterative solver that performs a dense matrix-vector multiplication in $O(N)$ complexity. It also outperforms a linear-complexity $\mathcal{H}^2$-based direct IE solver that does not minimize the rank and optimize the $\mathcal{H}^2$-matrix partition based on accuracy. In addition, the proposed method can be applied in an $\mathcal{H}^2$-matrix-based framework to further reduce the computational cost of a direct IE-based solution.

References

[1] K. Nabors and J. White, "FastCap: A multipole accelerated 3-D capacitance extraction program," *IEEE Trans. Integr. Circuits Syst., Comput.-Aided Design*, vol. 10, no. 11, pp. 1447–1459, Nov. 1991.

[2] W. Shi, J. Liu, N. Kakani, and T. Yu, "A fast hierarchical algorithm for 3-D capacitance extraction," *IEEE Trans. Integr. Circuits Syst., Comput.-Aided Design*, vol. 21, no. 3, pp. 330–336, Mar. 2002.

[3] S. Yan, V. Saren, and W. Shi, "Sparse transformations and preconditioners for hierarchical 3-D capacitance extraction with multiple dielectrics," in *Proc. 41st Annu. Design Autom. Conf.*, 2004, pp. 788–793.

[4] S. Kapur and D. E. Long, "IES3 : A fast integral equation solver for efficient 3-dimensional extraction," in *Proc. IEEE Int. Conf., Comput.-Aided Design Dig. Technol.*, Nov. 1997, pp. 448–455.

[5] J. R. Phillips and J. White, "A precorrected FFT method for capacitance extraction of complicated 3-D structures," in *Proc. IEEE Int. Conf. Comput.-Aided Design*, Mar. 1994, pp. 268–271.

[6] Y. C. Pan, W. C. Chew, and L. X. Wan, "A fast multipole-method based calculation of the capacitance matrix for multiple conductors above stratified dielectric media," *IEEE Trans. Microw. Theory Technol.*, vol. 49, no. 3, pp. 480–490, Mar. 2001.

[7] R. Jiang, Y.-H. Chang, and C. C.-P. Chen, "ICCAP: A linear time sparsification and reordering algorithm for 3D BEM capacitance extraction," *IEEE Trans. Microw. Theory Technol.*, vol. 54, no. 7, pp. 3060–3068, Jul. 2006.

[8] W. Chai, D. Jiao, and C. C. Koh, "A direct integral-equation solver of linear complexity for large-scale 3D capacitance and impedance extraction," in *Proc. 46th ACM/EDAC/IEEE Design Autom. Conf.*, Jul. 2009, pp. 752–757.

[9] W. Chai and D. Jiao, "Dense matrix inversion of linear complexity for integral-equation based large-scale 3-D capacitance extraction," *IEEE Trans. Microw. Theory Technol*, vol. 59, no. 10, pp. 2404–2421, Oct. 2011.

[10] W. Chai and D. Jiao, "An LU decomposition based direct integral equation solver of linear complexity and higher-order accuracy for large-scale interconnect extraction," *IEEE Trans. Adv. Packag.*, vol. 33, no. 4, pp. 794–803, Nov. 2010.

[11] S. Börm, "$\mathcal{H}^2$-matrices—multilevel methods for the approximation of integral operators," *Comput. Visual. Sci.*, vol. 7, no. 3, pp. 173–181, Oct. 2004.

[12] S. Börm and W. Hackbusch, "$\mathcal{H}^2$-matrix approximation of integral operators by interpolation," *Appl. Numer. Math.*, vol. 43. 2002, pp. 129–143.

[13] S. Börm, "$\mathcal{H}^2$-matrix arithmetics in linear complexity," in *Proc. Comput. Conf.*, 2006, pp. 1–28.

[14] S. Börm, L. Grasedyck, and W. Hackbusch, "Hierarchical matrices," in *Lecture Note 21 of the Max Planck Institute for Mathematics in the Sciences*. Princeton, NJ: Princeton Univ. Press, 2003.

[15] W. Hackbusch and B. Khoromskij, "A sparse matrix arithmetic based on $\mathcal{H}$-matrices. Part I: Introduction to $\mathcal{H}$-matrices," in *Proc. Comput. Conf.*, 1999, pp. 89–108.

[16] W. Hackbusch and B. Khoromskij, "A sparse matrix arithmetic. Part II: Application to multi-dimensional problems," *Computing*, vol. 64, pp. 21–47, Jan. 2000.

[17] S. Börm, "Introduction to hierarchical matrices with applications," *Eng. Anal. Boundary Elemen.*, vol. 27, no. 5, pp. 403–564, May 2003.

[18] L. Grasedyck, "Adaptive recompression of $\mathcal{H}$-matrices for BEM," *Computing*, vol. 74, no. 3, pp. 205–223, May 2005.

[19] M. Bebendorf, "Approximation of boundary element matrices," *Numer. Math.*, vol. 86, no. 4, pp. 565–589, 2000.

[20] C. F. V. Loan and G. H. Golub, *Matrix Comput.*. London, U.K.: Johns Hopkins Univ. Press, 1996.

Wenwen Chai received the B.S. degree from the University of Science and Technology of China, Hefei, China, the M.S. degree from Chinese Academy of Sciences, Beijing, China, and the Ph.D. degree from Purdue University, West Lafayette, IN, in 2004, 2007, and 2012, respectively, all in electrical engineering.

She was a Research Assistant with the On-Chip Electromagnetics Group, Purdue University, from 2007 to 2012. She joined Synopsys Inc. as a Senior R&D Engineer with the PrimeRail Group in 2012. Her current research interests include computational electromagnetics, high-performance VLSI CAD, and fast and high-capacity numerical methods.

Dr. Chai was the recipient of the IEEE Antennas and Propagation Society Doctoral Research Award for 2009 to 2010 and the Synopsys IG Division Individual Recognition Award in 2012 for her contribution to the performance improvement of the large-scale power grid analysis tool.

Dan Jiao (S'00–M'02–SM'06) received the Ph.D. degree in electrical engineering from the University of Illinois at Urbana-Champaign, Urbana, in 2001.

She was with the Technology Computer-Aided Design (CAD) Division, Intel Corporation, from 2001 to 2005, as a Senior CAD Engineer, Staff Engineer, and Senior Staff Engineer. In 2005, she joined the School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN, as an Assistant Professor, where she is currently a tenured Associate Professor. She has authored or co-authored over 170 papers in refereed journals and international conferences and authored two book chapters. Her current research interests include computational electromagnetics, high-frequency digital, analog, mixed-signal, and radio frequency (RF) integrated circuit (IC) design and analysis, high-performance VLSI CAD, modeling of microscale and nanoscale circuits, applied electromagnetics, fast and high-capacity numerical methods, fast time-domain analysis, scattering and antenna analysis, RF, microwave, and millimeter-wave circuits, wireless communication, and bioelectromagnetics.

Dr. Jiao was the recipient of the 2000 Raj Mittra Outstanding Research Award presented by the University of Illinois at Urbana-Champaign, the Intel Corporation's LTD Team Quality Award for her outstanding contribution to the development of the measurement capability and simulation tools for high-frequency on-chip crosstalk, the 2002 Intel Corporation's Components Research the Intel Hero Award (Intel-wide she was the tenth recipient) for the timely and accurate 2-D and 3-D full-wave simulations, the Intel Corporation's Technology CAD Divisional Achievement Award for the development of innovative full-wave solvers for high frequency IC design, the 2003 Intel Corporation's Logic Technology Development (LTD) Divisional Achievement Award in recognition of her work on the industry-leading BroadSpice modeling/simulation capability for designing high-speed microprocessors, packages, and circuit board, the 2004 Best Paper Award presented at the Intel Corporation's annual corporate-wide technology conference (Design and Test Technology Conference) for her work on generic broadband model of high-speed circuits, a 2006 Office of Naval Research (ONR) Award under the Young Investigator Program, the 2006 Jack and Cathie Kozik Faculty Start up Award (which recognizes an outstanding new faculty member of the School of Electrical and Computer Engineering, Purdue University), the 2008 National Science Foundation (NSF) CAREER Award, and the 2010 Ruth and Joel Spira Outstanding Teaching Award. She has been a reviewer of many IEEE journals and conferences. She is an Associate Editor of the IEEE TRANSACTIONS ON COMPONENTS, PACKAGING, AND MANUFACTURING TECHNOLOGY. She was among the 85 engineers selected throughout the nation for the National Academy of Engineering's 2011 US Frontiers of Engineering Symposium.