

Dense Matrix Inversion of Linear Complexity for Integral-Equation-Based Large-Scale 3-D Capacitance Extraction

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

Abstract—State-of-the-art integral-equation-based solvers rely on techniques that can perform a dense matrix–vector multiplication in linear complexity. We introduce the \mathcal{H}^2 matrix as a mathematical framework to enable a highly efficient computation of dense matrices. Under this mathematical framework, as yet, no linear complexity has been established for matrix inversion. In this work, we developed a matrix inverse of linear complexity to directly solve the dense system of linear equations for the 3-D capacitance extraction involving arbitrary geometry and nonuniform materials. We theoretically proved the existence of the \mathcal{H}^2 matrix representation of the inverse of the dense system matrix, and revealed the relationship between the block cluster tree of the original matrix and that of its inverse. We analyzed the complexity and the accuracy of the proposed inverse, and proved its linear complexity, as well as controlled accuracy. The proposed inverse-based direct solver has demonstrated clear advantages over state-of-the-art capacitance solvers such as FastCap and HiCap: with fast CPU time and modest memory consumption, and without sacrificing accuracy. It successfully inverts a dense matrix that involves more than one million unknowns associated with a large-scale on-chip 3-D interconnect embedded in inhomogeneous materials with fast CPU time and less than 5-GB memory.

Index Terms—Capacitance extraction, direct solver, \mathcal{H}^2 matrix, integral-equation-based methods, matrix inversion.

I. INTRODUCTION

INTEGRAL-EQUATION-BASED (IE-based) methods have been a popular choice in extracting the capacitive parameters of 3-D interconnects since they reduce the solution domain by one dimension, and they model an infinite domain without the need of introducing a truncation boundary condition. Compared to their partial-differential-equation-based counterparts, however, IE-based methods generally lead to dense systems of linear equations. Using a naïve direct method to solve a dense system takes $O(N^3)$ operations and requires $O(N^2)$ space, with N being the matrix size. When an iterative solver is used, the memory requirement remains the same, and the time complexity is $O(N_{\text{rhs}}N_{\text{it}}N^2)$, where N_{it} denotes

the total number of iterations required to reach convergence, and N_{rhs} is the number of right-hand sides. In state-of-the-art IE-based solvers [1]–[9], [22], fast multipole method and hierarchical algorithms were used to perform a matrix–vector multiplication in $O(N)$ complexity, thereby significantly reducing the complexity of iterative solvers; efficient preconditioners [8], [9] were also developed to reduce the number of iterations. In the limited work reported on the direct IE solutions [6], [10], [22], [24], [25], the best complexity is shown to be $O(N \log^\alpha N)$. No linear complexity has been achieved for general 3-D problems. Compared to iterative solvers, direct solvers have advantages when the number of iterations is large or the number of right-hand sides is large. A linear-complexity inverse-based direct solver has an additional advantage in memory compared to iterative solvers. Consider a system of N_c conductors. Using existing fast iterative solvers, even if each matrix solve is of linear complexity, to store the capacitance matrix one has to use $O(N_c^2)$ storage units. In contrast, with an inverse having linear complexity in both CPU time and memory consumption, the capacitance matrix can be stored in $O(N_c)$ units.

The contribution of this paper is the development of a linear-complexity inverse-based direct IE solver. To be specific, the inverse of a dense system matrix arising from a capacitance extraction problem is obtained in linear CPU time and memory consumption without sacrificing accuracy. Our solution hinges on the observation that the matrices resulting from an IE-based method, although dense, can be thought of as *data sparse*, i.e., they can be specified by few parameters. There exists a general mathematical framework, called the “Hierarchical (\mathcal{H}) Matrix” framework [10]–[12], which enables a highly compact representation and efficient numerical computation of dense matrices. Both storage requirements and matrix–vector multiplications using \mathcal{H} matrices are of complexity $O(N \log^\alpha N)$. \mathcal{H}^2 -matrices, which are a specialized subclass of hierarchical matrices, were later introduced in [13]–[16]. It was shown that the storage requirements and matrix–vector products are of complexity $O(N)$ for \mathcal{H}^2 -based computation of both quasi-static [10] and electrodynamic problems [17], [18] from small to tens of wavelengths. It was also shown that an \mathcal{H}^2 -based matrix–matrix multiplication can be performed in linear complexity [16]. The nested structure is the key difference between \mathcal{H} -matrices and \mathcal{H}^2 -matrices since it permits an efficient reuse of information across the entire hierarchy.

The \mathcal{H}^2 -matrix-based direct matrix solution of linear complexity has not been established in the literature. In this work, we developed an \mathcal{H}^2 -matrix-based inverse of linear complexity

Manuscript received January 29, 2011; accepted June 13, 2011. Date of publication July 29, 2011; date of current version October 12, 2011. This work was supported by the National Science Foundation (NSF) under Award 0747578 and Award 0702567.

The authors are with the School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47907 USA (e-mail: djiao@purdue.edu; wchai@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/TMTT.2011.2160964

for large-scale capacitance extraction. In [19], we outlined the basic idea of this work. In this paper, we complete the work from both theoretical and numerical perspectives. The significant extension over [19] is as follows.

First, we prove the existence of an \mathcal{H}^2 -matrix-based representation of the dense system matrix *as well as its inverse* for capacitance extraction involving arbitrary inhomogeneity and arbitrary geometry. We show that the \mathcal{H}^2 -based representation of the original matrix is error bounded, and the same is true for the \mathcal{H}^2 -based representation of its inverse. Moreover, we prove that the inverse and the original matrix share the same block cluster tree structure, and the cluster bases constructed from the original matrix can be used for the \mathcal{H}^2 -based representation of its inverse. This proof serves as a theoretical basis for developing \mathcal{H}^2 -matrix-based fast direct solutions of controlled accuracy for capacitance extraction.

Second, we show how to construct a block cluster tree to efficiently represent both the original matrix and its inverse for the capacitance extraction in inhomogeneous media.

Third, we present detailed linear-complexity algorithms in the proposed inverse and analyze their complexity. In [19], we only gave a very high-level picture of the algorithm, and the complexity analysis is only for the multiplications involved in the inverse procedure. In this work, we provide a complete inverse algorithm and its complexity analysis. To help better understand the proposed linear-complexity inverse, we use an analogy between a matrix–matrix multiplication and a matrix inverse to present the proposed algorithm since the \mathcal{H}^2 -based matrix–matrix multiplication has been shown to have a linear complexity [16]. We first make a comparison between a matrix inverse and a matrix–matrix multiplication to reveal their similarity, as well as their difference. We show that although the two operations share the same number of block matrix multiplications, there is a major difference that prevents one from directly using the linear-time matrix–matrix multiplication algorithm to achieve a linear complexity in inverse. The major difference is that in the level-by-level computation of the inverse, at each level, the computation is performed based on updated matrix blocks obtained from the computation at the previous level instead of the original matrix. In contrast, in the level-by-level computation of the matrix–matrix multiplication, at each level, the computation is always performed based on the original matrix, which is never updated. This difference would render the inverse complexity higher than linear if one does not address it properly. We then detail the algorithms in the proposed inverse that overcome this issue. In addition, we greatly enrich the section of numerical results.

This paper is organized as follows. In Section II, we derive the \mathcal{H}^2 -matrix-based representation of the dense system matrix resulting from capacitance extraction and show that this representation is error bounded. In addition, we prove the existence of the \mathcal{H}^2 representation of the inverse and reveal its relationship with the \mathcal{H}^2 representation of the original matrix. In Section III, we construct a block cluster tree for an efficient \mathcal{H}^2 -based representation of the dense system matrix and its inverse. In Section IV, we provide an overall procedure of the proposed direct solver. In Section V, we make a comparison between a matrix–matrix product and a matrix inverse, from

which one can clearly see the difference between these two. In Section VI, we detail the linear-complexity algorithms in the proposed inverse. In Section VII, we give numerical results to demonstrate the accuracy and linear complexity of the proposed direct IE solver for capacitance extraction. Comparisons with state-of-the-art capacitance solvers such as FastCap and HiCap are also presented. We conclude in Section VIII.

To help make this paper concise, in what follows, we do not repeat mathematics that can be referred to in the \mathcal{H}^2 -matrix literature. We only keep those mathematical definitions that are necessary for the completeness of this paper so that we can focus on the proposed new algorithms.

II. \mathcal{H}^2 MATRIX REPRESENTATION OF THE DENSE SYSTEM MATRIX AND ITS INVERSE FOR CAPACITANCE EXTRACTION

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], [19]:

$$\mathbf{G}q = v \quad (1)$$

where $\mathbf{G} = \begin{bmatrix} \mathbf{P}_{cc} & \mathbf{P}_{cd} \\ \mathbf{E}_{dc} & \mathbf{E}_{dd} \end{bmatrix}$, $q = \begin{bmatrix} q_c \\ q_d \end{bmatrix}$, 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 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

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

where a_i and a_j are the areas of panel S_i and S_j , respectively, g is the static Green's function, and ε_a and ε_b are the permittivity of two different materials. The diagonal entries of \mathbf{E}_{dd} are $e_{ij} = (\varepsilon_a + \varepsilon_b)/(2a_i\varepsilon_0)$.

In a uniform dielectric, (1) is reduced to

$$\mathbf{P}_{cc}q_c = v_c. \quad (3)$$

Next, we show that the dense system matrix \mathbf{G} shown in (1) can be represented by an \mathcal{H}^2 matrix with error well controlled. Moreover, the inverse of \mathbf{G} also has an \mathcal{H}^2 representation. Such a property holds true for any \mathbf{G} , i.e., IE-based capacitance extraction involving arbitrary geometry and inhomogeneity.

First we introduce the definitions of an \mathcal{H} matrix and an \mathcal{H}^2 matrix. An \mathcal{H}^2 matrix is generally associated with a strong admissibility condition [10, pp. 145]. To define a strong admissibility condition, we denote the full index set of all the panels by $\mathcal{I} := \{1, 2, \dots, N\}$, where N is the total number of panels, and hence, unknowns. Considering two subsets t and s of \mathcal{I} , the strong admissibility condition is defined as

$$\max\{\text{diam}(\Omega_t), \text{diam}(\Omega_s)\} \leq \eta \text{dist}(\Omega_t, \Omega_s) \quad (4)$$

where Ω_t and Ω_s are the supports of the union of all the panels in t and s , respectively, $\text{diam}(\cdot)$ is the Euclidean diameter of a set, $\text{dist}(\cdot, \cdot)$ is the Euclidean distance between two sets, and η is a

positive parameter. If subsets t and s satisfy (4), they are admissible, in other words, they are well separated; otherwise, they are inadmissible. Generally, it is not practical to directly measure the Euclidean diameter and Euclidean distance. We thus use an axis-parallel bounding box $Q_t \supseteq \Omega_t$, which is the tensor product of intervals [10, pp 46–48], to represent the support of the union of all the panels in t .

Denoting the matrix block formed by t and s by $\mathbf{G}^{t,s}$, if all the blocks $\mathbf{G}^{t,s}$ formed by the admissible (t, s) in \mathbf{G} can be represented by a low-rank matrix, \mathbf{G} is an \mathcal{H} matrix. In other words, if \mathbf{G} possesses the following property:

$$\mathbf{G} \in \mathbb{R}^{\#I \times \#I} : \mathbf{G}^{t,s} \text{ is low rank for all admissible } (t, s) \quad (5)$$

it is an \mathcal{H} matrix.

If \mathbf{G} can be further written as a factorized form

$$\begin{aligned} \tilde{\mathbf{G}}^{t,s} &:= \mathbf{V}^t \mathbf{S}^{t,s} \mathbf{V}^{s\top} \\ \mathbf{V}^t &\in \mathbb{R}^{\#t \times k} \\ \mathbf{S}^{t,s} &\in \mathbb{R}^{k \times k} \\ \mathbf{V}^s &\in \mathbb{R}^{\#s \times k} \end{aligned} \quad (6)$$

where \mathbf{V}^t is nested, then \mathbf{G} is an \mathcal{H}^2 matrix. In (6), \mathbf{V}^t is called a cluster basis, $\mathbf{S}^{t,s}$ is called a coupling matrix, k is the rank of \mathbf{V}^t , and “#” denotes the cardinality of a set. The nested property of \mathbf{V}^t enables $O(N)$ storage of a dense matrix and $O(N)$ matrix–vector multiplication [10, p. 146].

A. \mathcal{H}^2 -Matrix Representation of \mathbf{G} With Error Well Controlled

1) \mathcal{H}^2 -Matrix Representation of \mathbf{G} : If two subsets t and s of \mathcal{I} satisfy the strong admissibility condition (4), the original kernel function $g(r_i, r_j)$ in (2) can be replaced by a degenerate approximation

$$\tilde{g}^{t,s}(r_i, r_j) = \sum_{v \in K^t} \sum_{\mu \in K^s} g(\xi_v^t, \xi_\mu^s) L_v^t(r_i) L_\mu^s(r_j) \quad (7)$$

where $K := \{v \in N^d : v_i \leq p \text{ for all } i \in \{1, \dots, d\}\} = \{1, \dots, p\}^d$, $d = 1, 2, 3$, for 1-D, 2-D, and 3-D problems, respectively; p is the number of interpolation points; $(\xi_v^t)_{v \in K^t}$ and $(\xi_\mu^s)_{\mu \in K^s}$ are two families of interpolation points, respectively, in t and s ; and $(L_v^t)_{v \in K^t}$ and $(L_\mu^s)_{\mu \in K^s}$ are the corresponding Lagrange polynomials. The interpolation in (7) is performed on the axis-parallel bounding boxes Q_t and Q_s .

With (7), the double integrals in (2) are separated into two single integrals

$$\begin{aligned} \tilde{\mathbf{P}}_{ij}^{t,s} &:= \sum_{v \in K^t} \sum_{\mu \in K^s} \frac{1}{a_i} \frac{1}{a_j} g(\xi_v^t, \xi_\mu^s) \int_{S_i} L_v^t(r_i) dr_i \\ &\cdot \int_{S_j} L_\mu^s(r_j) dr_j \end{aligned} \quad (8)$$

$$\begin{aligned} \tilde{\mathbf{E}}_{ij}^{t,s} &= \sum_{v \in K^t} \sum_{\mu \in K^s} \frac{1}{a_i} \frac{1}{a_j} (\varepsilon_a - \varepsilon_b) \frac{\partial g(\xi_v^t, \xi_\mu^s)}{\partial n_a} \int_{S_i} L_v^t(r_i) dr_i \\ &\cdot \int_{S_j} L_\mu^s(r_j) dr_j. \end{aligned} \quad (9)$$

Hence, the submatrix $\tilde{\mathbf{G}}^{t,s}$ can be written in a factorized form as

$$\begin{aligned} \tilde{\mathbf{G}}^{t,s} &:= \mathbf{V}^t \mathbf{S}^{t,s} \mathbf{V}^{s\top} & \mathbf{V}^t &\in \mathbb{R}^{\#t \times \#K^t} \\ \mathbf{S}^{t,s} &\in \mathbb{R}^{\#K^t \times \#K^s} & \mathbf{V}^s &\in \mathbb{R}^{\#s \times \#K^s} \end{aligned} \quad (10)$$

where

$$\begin{aligned} \mathbf{V}_{iv}^t &= \int_{S_i} L_v^t(r_i) dr_i & \mathbf{V}_{j\mu}^s &= \int_{S_j} L_\mu^s(r_j) dr_j \\ \mathbf{S}_{v\mu}^{t,s} &= \begin{cases} \frac{g(\xi_v^t, \xi_\mu^s)}{(a_i a_j)}, & t \text{ contains conductor panels} \\ \frac{(\varepsilon_a - \varepsilon_b) \partial g(\xi_v^t, \xi_\mu^s)}{(a_i a_j) \partial n_a}, & t \text{ contains dielectric panels} \end{cases} \end{aligned} \quad (11)$$

for $i \in t$, $j \in s$, $v \in K^t$ and $\mu \in K^s$.

If we use the same space of polynomials for all clusters, then \mathbf{V}^t is nested. To explain, consider a set t' , which is a subset of t , $L_v^t(r)$ in (11) can be written as

$$L_v^t(r) = \sum_{v' \in K^{t'}} \mathbf{T}_{v'v}^{t'} L_{v'}^{t'}(r) \quad (12)$$

where

$$\mathbf{T}_{v'v}^{t'} = L_v^t(\xi_{v'}^{t'}). \quad (13)$$

As a result, \mathbf{V}_{iv}^t in (11) can be written as

$$\begin{aligned} \mathbf{V}_{iv}^t &= \int_{S_i} L_v^t(\vec{r}) dr \\ &= \sum_{v' \in K^{t'}} \mathbf{T}_{v'v}^{t'} \int_{S_i} L_{v'}^{t'}(\vec{r}) dr \\ &= \sum_{v' \in K^{t'}} \mathbf{T}_{v'v}^{t'} \mathbf{V}_{iv'}^{t'} \\ &= (\mathbf{V}^{t'} \mathbf{T}^{t'})_{iv} \end{aligned} \quad (14)$$

where $\mathbf{T}^{t'} \in \mathbb{R}^{\#K^{t'} \times \#K^{t'}}$ is called a transfer matrix for the subset t' . Hence, assuming that the set t is the union of two subsets t_1 and t_2 , we have

$$\mathbf{V}^t = \begin{pmatrix} \mathbf{V}^{t_1} \mathbf{T}^{t_1} \\ \mathbf{V}^{t_2} \mathbf{T}^{t_2} \end{pmatrix} = \begin{pmatrix} \mathbf{V}^{t_1} & \\ & \mathbf{V}^{t_2} \end{pmatrix} \begin{pmatrix} \mathbf{T}^{t_1} \\ \mathbf{T}^{t_2} \end{pmatrix}. \quad (15)$$

Thus, \mathbf{V}^t is nested.

From (10) and (15), we prove that the dense system matrix \mathbf{G} for capacitance extraction can be represented by an \mathcal{H}^2 matrix. In Section II-A.2, we show that such a representation is error bounded.

2) *Error Bound*: Following the derivation in [18], if the admissibility condition given in (4) is satisfied, the error of (7) is bounded by

$$\begin{aligned} \|g(r, r') - \tilde{g}^{(t,s)}(r, r')\|_{\infty, Q_t \times Q_s} \\ \leq \frac{4ed}{\pi} (\Lambda_p)^{2d} p \frac{1}{\text{dist}(Q_t, Q_s)} [1 + \sqrt{2}\eta] \left[1 + \frac{\sqrt{2}}{\eta}\right]^{-p} \end{aligned} \quad (16)$$

where Λ_p is a constant related to p and the interpolation scheme. Clearly, exponential convergence with respect to p can be obtained irrespective of the choice of η . Since $\mathbf{G}_{ij}^{(t,s)}$ is proportional to $1/\text{dist}(Q_t, Q_s)$, the relative error becomes a constant related to η and p . The smaller the η , the smaller the error; the larger the p , the smaller the error. In addition, all block entries represented by (10) can be kept to the same order of accuracy across the levels of a block cluster tree.

B. \mathcal{H}^2 -Matrix Representation of \mathbf{G}^{-1}

In this section, to help better understand the existence of the \mathcal{H}^2 -matrix representation of \mathbf{G}^{-1} , we provide a mathematical proof.

Consider a 3-D problem involving arbitrarily shaped conductors embedded in nonuniform materials. The electrostatic phenomena in such a problem are governed by Poisson's equation

$$-\nabla \cdot (\varepsilon \nabla v) = \rho_s \quad (17)$$

where v is electric potential and ρ_s is charge density. By using a differencing scheme to discretize the space derivatives in Poisson's equations, like what is done in a partial differential-equation-based solution of (17), we obtain the following system of equations:

$$\mathbf{C}V = Q \quad (18)$$

where V is a vector consisting of the electric potential at each discretized point in the 3-D computational domain, and Q is a vector containing the charge density at each discretized point. Due to the nature of the partial differential operator, the charge density at each discretized point only needs to be evaluated from the electric potentials that are adjacent to the point. As a result, in each row of \mathbf{C} , there are only a few nonzero elements, which are contributed by the electric potentials close to the point corresponding to the row index. Thus, \mathbf{C} in (18) is a sparse matrix, and also its blocks satisfying admissibility condition (4) are all zero.

Each row of (1) states that the total electric potential at one point in space is the superposition of the electric potential generated by all of the discrete charges. Therefore, if (1) is formulated for all of the discretized points in a 3-D volumetric domain, then \mathbf{G}^{-1} is nothing but \mathbf{C} , and hence, a sparse matrix.

However, due to a surface integral based formulation, in (1), the right-hand side v is not the complete V ; instead, it is a subset of V , which only consists of the electric potential on the conducting surface and that on the dielectric-dielectric interface. Therefore, \mathbf{G}^{-1} is not directly \mathbf{C} in (18). However, there exists a relationship between \mathbf{G}^{-1} and \mathbf{C} , which dictates the existence of the \mathcal{H}^2 -matrix representation of \mathbf{G}^{-1} . To see this relationship, we rewrite (18) as

$$\begin{bmatrix} \mathbf{C}_{11} & \mathbf{C}_{12} \\ \mathbf{C}_{21} & \mathbf{C}_{22} \end{bmatrix} \begin{Bmatrix} v \\ v_{\text{else}} \end{Bmatrix} = \begin{Bmatrix} q \\ 0 \end{Bmatrix} \quad (19)$$

where v and q are the same as those in (1), and v_{else} denotes the electric potential elsewhere, which is not associated with the conducting surfaces and dielectric interfaces. Since the charge density is zero in a purely dielectric region, the right-hand side

corresponding to the second row in (19) is zero. From (19), we immediately obtain

$$(\mathbf{C}_{11} - \mathbf{C}_{12}\mathbf{C}_{22}^{-1}\mathbf{C}_{21})v = q. \quad (20)$$

Comparing (20) to (1), it is clear that

$$\mathbf{G}^{-1} = \mathbf{C}_{11} - \mathbf{C}_{12}\mathbf{C}_{22}^{-1}\mathbf{C}_{21}. \quad (21)$$

The second row of (19), $\mathbf{C}_{22}v_{\text{else}} = -\mathbf{C}_{21}v$, is what is traditionally solved by a partial differential-equation-based method: solving v_{else} subject to boundary condition v . It is clear that \mathbf{C}_{22}^{-1} is the inverse of the matrix resulting from the discretization of a Poisson's operator. It is proven in [23] that the inverse of the matrix resulting from the discretization of an elliptic partial differential operator has an \mathcal{H} -matrix representation. Therefore, \mathbf{C}_{22}^{-1} also has an \mathcal{H} -matrix representation, and hence, an \mathcal{H}^2 -matrix representation (an \mathcal{H} -matrix representation can be converted to an \mathcal{H}^2 -matrix representation [10]). This can also be seen clearly from the fact that \mathbf{C}_{22}^{-1} is nothing but \mathbf{G}_{22} , the \mathbf{G} matrix whose row/column dimension is the same as the length of v_{else} , and each column of \mathbf{G}_{22} represents the electric potential v_{else} generated by one charge configuration (the $-\mathbf{C}_{21}v$ is in fact an equivalent charge vector). The \mathbf{G} matrix's \mathcal{H}^2 matrix representation has already been shown in Section II-A. Therefore, \mathbf{C}_{22}^{-1} has an \mathcal{H}^2 matrix representation.

To prove the existence of the \mathcal{H}^2 -matrix representation of \mathbf{G}^{-1} , we need to prove that all the blocks $(\mathbf{G}^{-1})^{t,s}$ formed by the admissible (t, s) in \mathbf{G}^{-1} can be represented by a factorized low-rank form shown in (6).

Consider a (t, s) block in \mathbf{G}^{-1} that satisfies the admissibility condition (4). Since unknowns in subset t and those in s are well separated based on the definition of the admissibility condition, we have

$$\mathbf{C}_{11}^{(t,s)} = 0 \quad (22)$$

because \mathbf{C}_{11} is a sparse matrix whose nonzero elements only appear in the close-interaction blocks. Therefore, from (21),

$$(\mathbf{G}^{-1})^{(t,s)} = -(\mathbf{C}_{12}\mathbf{C}_{22}^{-1}\mathbf{C}_{21})^{(t,s)}. \quad (23)$$

The (t, s) block of $(\mathbf{C}_{12}\mathbf{C}_{22}^{-1}\mathbf{C}_{21})$ can be evaluated as

$$(\mathbf{C}_{12}\mathbf{C}_{22}^{-1}\mathbf{C}_{21})^{(t,s)} = (\mathbf{C}_{12})^{(t,t')}(\mathbf{C}_{22}^{-1})^{(t',s')}(\mathbf{C}_{21})^{(s',s)} \quad (24)$$

where t' denotes the subset that is physically close to t , s' denotes the subset that is physically close to s . As shown in Fig. 1, $(\mathbf{C}_{12})^{(t,t')}$ denotes the *nonzero* block in \mathbf{C}_{12} that occupies rows corresponding to subset t , and $(\mathbf{C}_{21})^{(s',s)}$ denotes the *nonzero* block in \mathbf{C}_{21} that has columns corresponding to subset s . In (24), we only need to consider $(\mathbf{C}_{12})^{(t,t')}$ among all of the $(\mathbf{C}_{12})^{(t,i)}$ ($i = 1, 2, \dots$) blocks because all the other blocks are zero since the unknowns in the corresponding two subsets are well separated from each other. This is the same reason why we only need to consider the $(\mathbf{C}_{21})^{(s',s)}$ block in \mathbf{C}_{21} . As a result, among all the blocks in \mathbf{C}_{22}^{-1} , only the (t', s') block participates in the computation of $\mathbf{C}_{12}\mathbf{C}_{22}^{-1}\mathbf{C}_{21}$, as illustrated in Fig. 1. Since the subset t' is close to subset t , subset s' is adjacent to subset s , and subsets t and s are well separated; the subset t' and subset s' also satisfy the admissibility condition (4). Thus, $(\mathbf{C}_{22}^{-1})^{(t',s')}$

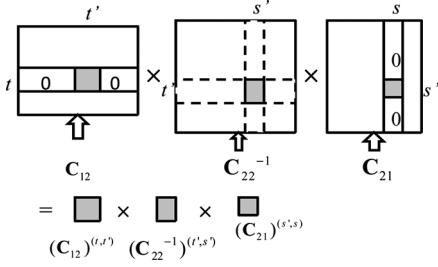


Fig. 1. Illustration of the actual operation involved in $\mathbf{C}_{12}\mathbf{C}_{22}^{-1}\mathbf{C}_{21}$.

has an \mathcal{H}^2 representation since $(\mathbf{C}_{22}^{-1})^{(t',s')}$ is $\mathbf{G}_{22}^{(t',s')}$. By using the \mathcal{H}^2 representation of the admissible block $\mathbf{G}_{22}^{(t',s')}$, we have

$$\begin{aligned} & (\mathbf{C}_{12}\mathbf{C}_{22}^{-1}\mathbf{C}_{21})^{(t,s)} \\ &= (\mathbf{C}_{12})^{(t,t')} \mathbf{V}^{\#t' \times k} \mathbf{S}^{k \times k} (\mathbf{V}^T)^{k \times \#s'} (\mathbf{C}_{21})^{(s',s)} \\ &= \tilde{\mathbf{V}}^{\#t \times k} \mathbf{S}^{k \times k} (\tilde{\mathbf{V}}^T)^{k \times \#s}. \end{aligned} \quad (25)$$

Thus, from (23) and (25), we prove that $(\mathbf{G}^{-1})^{t,s}$ has an \mathcal{H}^2 matrix representation. Since (t, s) is an arbitrary admissible block, we conclude that, for all the admissible blocks in \mathbf{G}^{-1} , there exists an \mathcal{H}^2 representation. With that, we prove the existence of \mathcal{H}^2 representation for \mathbf{G}^{-1} .

Two important findings can be identified from the above proof. First, \mathbf{G} and \mathbf{G}^{-1} share the same block cluster tree structure in common. A block cluster tree determines which matrix block has an \mathcal{H}^2 form, which is a full matrix. As can be seen from the above proof, given an admissibility condition (4), if a block is admissible in \mathbf{G} , it must also be admissible in \mathbf{G}^{-1} (i.e., has a factorized low rank form); if a block is inadmissible in \mathbf{G} , it must also be inadmissible in \mathbf{G}^{-1} . Therefore, \mathbf{G} and \mathbf{G}^{-1} share the same block cluster tree structure. In addition, they share the same rank distribution as can be seen from (25). The second finding is that the same cluster basis constructed from the original matrix can be used to represent its inverse as can be seen from (25). If the first-order differencing scheme is used to discretize Poisson's equations, \mathbf{C}_{21} and \mathbf{C}_{12} are, in fact, diagonal matrices. For nondiagonal \mathbf{C}_{21} and \mathbf{C}_{12} , the $\tilde{\mathbf{V}}$ in (25) can always be spanned in the space of \mathbf{V} as the following:

$$\tilde{\mathbf{V}}^{\#t \times k} = \mathbf{V}^{\#t \times k} \mathbf{Z}^{k \times k}$$

where

$$\mathbf{Z}^{k \times k} = [(\mathbf{V}^{\#t \times k})^T \mathbf{V}^{\#t \times k}]^{-1} (\mathbf{V}^{\#t \times k})^T (\mathbf{C}_{12})^{(t,t')} \mathbf{V}^{\#t' \times k}.$$

Thus, with \mathbf{V} being the cluster basis of the inverse, the only difference is that the coupling matrix will be modified correspondingly from that in (25) to $\mathbf{ZS}^{k \times k} \mathbf{Z}^T$. This is similar to the fact that given a set of cluster bases, one can always orthogonalize it to construct a new set of cluster bases without losing accuracy.

III. BLOCK CLUSTER TREE CONSTRUCTION FOR EFFICIENT STORAGE AND PROCESSING OF \mathcal{H}^2 -BASED \mathbf{G} AND \mathbf{G}^{-1}

In this section, we show how to construct a block cluster tree for the capacitance extraction problem. A block cluster tree is a

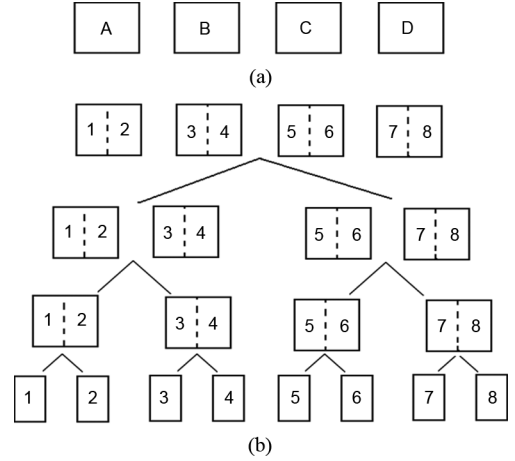


Fig. 2. (a) Example of a structure having four conductors. (b) Resultant cluster tree.

tree structure that can be used to efficiently capture the nested hierarchical dependence present in an \mathcal{H}^2 matrix [10, pp. 13–15]. Here, special care needs to be taken to make the \mathcal{H}^2 -based representation of \mathbf{G} and \mathbf{G}^{-1} efficient for capacitance extraction.

A. Block Cluster Tree Construction for \mathcal{H}^2 -Based \mathbf{G}

To make the explanation clear, we use a simple example to show the procedure of constructing a block cluster tree without loss of generality of the procedure. Consider a capacitance system made of four conductors, as shown in Fig. 2(a). We discretize each conductor into two panels, resulting in a panel set of $\mathcal{I} := \{1, 2, \dots, N\}$, where N is 8 in this example. We start from \mathcal{I} and split it into two subsets, as shown in Fig. 2(b). We continue to split until the number of panels involved in each subset is less than or equal to *leafsize*, which is a parameter to control the tree depth. For the specific example shown in Fig. 2(a), *leafsize* is 1. As a result, we generate a cluster tree, as shown in Fig. 2(b). The cluster tree constructed for panel set \mathcal{I} is denoted by $T_{\mathcal{I}}$. All the nodes of the tree are called as clusters. The full panel set \mathcal{I} is called the root cluster, denoted by $\text{Root}(T_{\mathcal{I}})$. Clusters with indices no more than *leafsize* are leaves. The set of leaves of $T_{\mathcal{I}}$ is denoted by $\mathcal{L}_{\mathcal{I}}$. Each nonleaf cluster has two children in our tree construction.

The block cluster tree is recursively constructed from cluster trees $T_{\mathcal{I}}$ and $T_{\mathcal{I}}$ and a given admissibility condition, the process of which is shown in Fig. 3. We start from $\text{Root}(T_{\mathcal{I}})$ and $\text{Root}(T_{\mathcal{I}})$, and test the admissibility condition between clusters $t \in T_{\mathcal{I}}$ and $s \in T_{\mathcal{I}}$ level by level. Once two clusters t and s are found to be admissible based on (4), a cross link is formed between them, which is called an admissible link. Once two clusters are linked, we do not check the admissibility condition for the combination of their children. If clusters t and s are both leaf clusters, but not admissible, they are also linked. For example, cluster $\{1\}$ and cluster $\{1\}$, shown in Fig. 3. This link is called an inadmissible link.

The aforementioned procedure results in a block cluster tree. Each link represents a leaf block cluster. The block cluster tree can be mapped to a matrix structure shown in Fig. 4. Each leaf

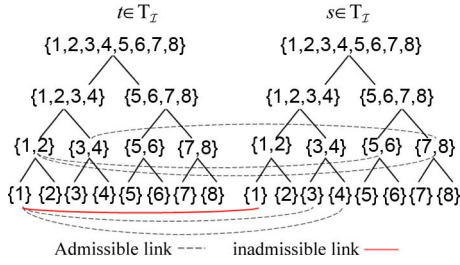


Fig. 3. Construction of a block cluster tree.

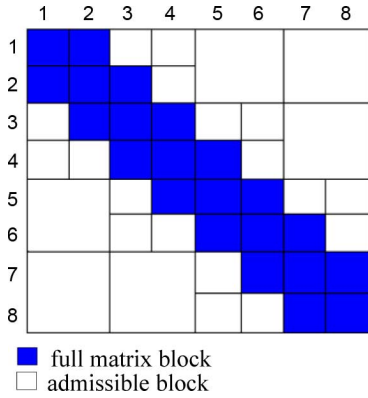
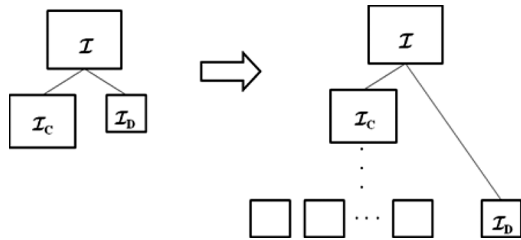
Fig. 4. \mathcal{H}^2 -matrix structure.

Fig. 5. Illustration of the treatment of the unbalanced case encountered in nonuniform dielectrics.

block cluster corresponds to a matrix block. The unshaded matrix blocks are admissible blocks in which the \mathcal{H}^2 -matrix-based representation is used; the shaded ones are inadmissible blocks in which a full matrix representation is employed.

Special treatment is required for structures involving multiple dielectrics. After discretizing the structure, the whole set that includes all the panels is divided into two subsets. One includes all the conductor panels, and the other includes all the dielectric panels, as shown in Fig. 5. The conductor set is denoted by \mathcal{I}_C , and the dielectric set is denoted by \mathcal{I}_D . If the two subsets are almost balanced, we can directly use the procedure above to construct the block cluster tree. If not, for example, if the number of conductor panels is much larger than that of dielectric panels, the subset \mathcal{I}_D constructed for dielectric panels is pushed down to the level where the size of clusters in \mathcal{I}_C is almost the same as that in \mathcal{I}_D . We then start to check the admissibility condition from that level. By doing so, the \mathcal{H}^2 -based representation of \mathbf{G} can be made more efficient.

B. Block Cluster Tree Construction for \mathcal{H}^2 -Based \mathbf{G}^{-1}

As proven in Section II-B, \mathbf{G}^{-1} is an \mathcal{H}^2 matrix, and also, has the same block cluster tree as \mathbf{G} . Thus, using the \mathcal{H}^2 tree of \mathbf{G} to represent that of \mathbf{G}^{-1} is theoretically rigorous for the integral operator encountered in the capacitance extraction.

IV. OVERALL PROCEDURE

In this section, we give the overall procedure of the proposed linear-complexity direct solver for capacitance extraction.

First, we introduce the concepts, notations, and parameters that are used throughout this paper.

- For each cluster $t \in T_{\mathcal{I}}$, the cardinality of the sets $\{s \in T_{\mathcal{I}} : (t, s) \in T_{\mathcal{I} \times \mathcal{I}}\}$ and $\{t \in T_{\mathcal{I}} : (t, s) \in T_{\mathcal{I} \times \mathcal{I}}\}$ is bounded by a constant C_{sp} [22, p. 124]. Graphically, C_{sp} is the maximum number of links that can be formed by a cluster at each level of a block cluster tree, as shown in Fig. 3.
- Each nonleaf cluster t has two child nodes.
- Each nonleaf block b has four children blocks.
- The rank of $\mathbf{V} = (\mathbf{V}^t)_{t \in T_{\mathcal{I}}}$ is denoted by k .
- The parameter *leafsize* is denoted by n_{\min} , and $\#t \leq n_{\min}$ if $t \in \mathcal{L}_{\mathcal{I}}$.
- $k_1 = \max(n_{\min}, k)$.

There are three steps in the proposed direct solver. At the first step, to enable linear-time matrix inversion, we orthogonalize cluster basis \mathbf{V}^t while still preserving the nested property of \mathbf{V}^t . Mathematically, the new basis $\tilde{\mathbf{V}}^t$ should satisfy the following two properties:

$$(\tilde{\mathbf{V}}^t)^T \tilde{\mathbf{V}}^t = \mathbf{I} \quad (26)$$

and

$$\tilde{\mathbf{V}}^t = \begin{bmatrix} \tilde{\mathbf{V}}^{t1} \tilde{\mathbf{T}}^{t1} \\ \tilde{\mathbf{V}}^{t2} \tilde{\mathbf{T}}^{t2} \end{bmatrix} \quad (27)$$

where $t_1, t_2 \in \text{children}(t)$. We employ the method in [14, pp. 254–258] to construct orthogonal bases $\tilde{\mathbf{V}}^t$, which is shown to have a linear complexity.

To give an example on how the orthogonalization helps achieve a linear complexity, consider one multiplication $\mathbf{G}^{b1} \times \mathbf{G}^{b2} \rightarrow \mathbf{G}^b$ involved in the inverse procedure, where $\mathbf{G}^{b1} = \tilde{\mathbf{V}}^t \mathbf{S}^{b1} \tilde{\mathbf{V}}^{sT}$ and $\mathbf{G}^{b2} = \tilde{\mathbf{V}}^s \mathbf{S}^{b2} \tilde{\mathbf{V}}^{rT}$, and $b = (t, r)$ is an admissible block in the inverse. Then,

$$\mathbf{G}^{b1} \times \mathbf{G}^{b2} = \tilde{\mathbf{V}}^t \mathbf{S}^{b1} \tilde{\mathbf{V}}^{sT} \times \tilde{\mathbf{V}}^s \mathbf{S}^{b2} \tilde{\mathbf{V}}^{rT}. \quad (28)$$

Since $\tilde{\mathbf{V}}$ is orthogonalized, we have

$$\mathbf{G}^{b1} \times \mathbf{G}^{b2} = \tilde{\mathbf{V}}^t \mathbf{S}^{b1} \mathbf{I} \mathbf{S}^{b2} \tilde{\mathbf{V}}^{rT} = \tilde{\mathbf{V}}^t (\mathbf{S}^{b1} \mathbf{S}^{b2}) \tilde{\mathbf{V}}^{rT}. \quad (29)$$

Thus, the multiplication cost becomes the cost of multiplying two coupling matrices \mathbf{S}^{b1} and \mathbf{S}^{b2} , each of which is a k by k matrix. Hence, the complexity of computing $\mathbf{G}^{b1} \times \mathbf{G}^{b2} \rightarrow \mathbf{G}^b$ is made $O(k^3)$, which is independent of the row dimension ($\#t$) and the column dimension ($\#r$) of \mathbf{G}^b . Notice that an \mathcal{H}^2 matrix is stored in the format of the cluster basis $\tilde{\mathbf{V}}$ and the coupling

underlined \mathbf{G} blocks in (32) are different from those in the original \mathbf{G} .

B. Matrix–Matrix Multiplication

Similar to matrix inverse, a matrix–matrix multiplication $\mathbf{G} \times \mathbf{G}$ can be recursively obtained from

$$\mathbf{G} \times \mathbf{G} = \begin{bmatrix} \mathbf{G}_{11} \times \mathbf{G}_{11} + \mathbf{G}_{12} \times \mathbf{G}_{21} & \mathbf{G}_{11} \times \mathbf{G}_{12} + \mathbf{G}_{12} \times \mathbf{G}_{22} \\ \mathbf{G}_{21} \times \mathbf{G}_{11} + \mathbf{G}_{22} \times \mathbf{G}_{21} & \mathbf{G}_{21} \times \mathbf{G}_{12} + \mathbf{G}_{22} \times \mathbf{G}_{22} \end{bmatrix} \quad (33)$$

which can be realized by the following pseudocode:

Procedure \mathcal{H}^2 -multiplication (\mathbf{G}, \mathbf{X}) (\mathbf{G} is input matrix,
 \mathbf{X} is output)

If matrix \mathbf{G} is a nonleaf matrix block

\mathcal{H}^2 -multiplication ($\mathbf{G}_{11}, \mathbf{X}_{11}$)
 $\mathbf{G}_{21} \times \mathbf{G}_{11} \rightarrow \mathbf{X}_{21}, \mathbf{G}_{11} \times \mathbf{G}_{12} \rightarrow \mathbf{X}_{12},$
 $\mathbf{X}_{11} + (\mathbf{G}_{12} \times \mathbf{G}_{21}) \rightarrow \mathbf{X}_{11},$
 \mathcal{H}^2 -multiplication ($\mathbf{G}_{22}, \mathbf{X}_{22}$)
 $\mathbf{X}_{21} + \mathbf{G}_{22} \times \mathbf{G}_{21} \rightarrow \mathbf{X}_{21}, \mathbf{X}_{12} + \mathbf{G}_{12} \times \mathbf{G}_{22} \rightarrow \mathbf{X}_{12},$
 $\mathbf{X}_{22} + (\mathbf{G}_{21} \times \mathbf{G}_{12}) \rightarrow \mathbf{X}_{22},$

else

DirectMultiply(\mathbf{G}) (normal full matrix multiplication)

(34)

C. Comparison

Comparing (32) with (34), it can be seen that the total number of block multiplications involved in a matrix inverse is exactly the same as that involved in a matrix–matrix multiplication; in addition, only a half number of additions in the matrix–matrix multiplication are involved in the inverse. In [16], it is shown that an \mathcal{H}^2 -based matrix–matrix multiplication can be performed in linear complexity. Apparently, the inverse can also be obtained in linear complexity using the \mathcal{H}^2 -based matrix–matrix multiplication algorithm. However, there exists a major difference between these two operations, which prevents one from directly using the matrix–matrix multiplication algorithm to achieve a linear-complexity inverse.

The major difference is that in the level-by-level computation of the inverse, at each level, the matrix blocks in \mathbf{G} are updated by their counterparts in \mathbf{G}^{-1} . Thus, one has to use updated matrix blocks to perform computation as highlighted by the underlined \mathbf{G} in (32). In contrast, in the level-by-level computation of the matrix–matrix multiplication, at each level, one always uses the original \mathbf{G} to perform computation. Once the product is computed, it will be stored in the corresponding target block in \mathbf{X} , as can be seen from (34), and never be used again in the following computations. Unlike (32), in (34), none of the \mathbf{G} is underlined, i.e., all of them come from the original matrix.

This major difference does not cause any difference in operation counts if one performs a conventional matrix inverse or matrix–matrix multiplication that has a cubic complexity. However, this difference leads to a significant difference in

devising a linear-complexity algorithm. The reasons are given below.

The linear-complexity matrix–matrix multiplication is achieved by a matrix forward transformation algorithm, a matrix backward transformation algorithm, and a recursive multiplication algorithm, as shown in [16, p. 21, Algorithm 10]. The matrix forward transformation used in the linear-time matrix–matrix multiplication cannot be used for inverse in the same way because in the inverse procedure, the matrix blocks in \mathbf{G} are kept updated in the level-by-level computation. The matrix forward transformation [16, p. 13, Algorithm 4] is used to prepare an auxiliary admissible block form of each block in \mathbf{A} and \mathbf{B} , i.e., $\check{\mathbf{S}}_{\mathbf{A}}$ and $\check{\mathbf{S}}_{\mathbf{B}}$. It is applicable to a matrix–matrix multiplication because all the matrix blocks involved in the multiplication are from the original matrix. They are never updated, and hence, a collected admissible block form $\check{\mathbf{S}}$ can be prepared in advance and can be directly used in the “RecursiveMultiply” function for the recursive multiplication. However, for inverse, the blocks at each level are kept updated and then are used to update other blocks, and hence, it is not possible to use the forward transformation to prepare the auxiliary admissible block forms ahead of the recursive inverse procedure.

A block matrix multiplication, when the target product block b is a nonleaf block, may generate a product that has an auxiliary admissible block form, i.e., $\check{\mathbf{S}}_b^C$ as shown in [16, p. 21, Algorithm 9]. To get the real matrix in b , $\check{\mathbf{S}}_b^C$ should be split to b ’s leaf blocks. However, since $\check{\mathbf{S}}_b^C$ is never involved in the subsequent computations in the matrix–matrix multiplication, it can be stored in the nonleaf block without being split immediately. After the matrix–matrix multiplication is done, a backward transformation [16, p. 14, Algorithm 5] can be used to split each $\check{\mathbf{S}}_b^C$ to the leaf blocks. Such a backward transformation, however, cannot be employed in the same way in the inverse procedure either. This is because in the inverse, $\check{\mathbf{S}}_b^C$ has to be used in the subsequent computations. We cannot wait until the inverse is done to process it. A straightforward way to overcome this problem is to split $\check{\mathbf{S}}_b^C$ to b ’s leaf blocks immediately after it is generated. However, this would, in general, result in a complexity greater than linear. Thus, one has to do it properly.

If the two essential operations, matrix forward transformation and matrix backward transformation, cannot be used in the same way in the inverse, each block matrix multiplication cannot be done in constant time. For example, when we do the block matrix multiplication based on [16, Algorithm 7], without the preparation of auxiliary admissible matrix $\check{\mathbf{S}}$, the cost for directly computing a block matrix multiplication would not be $O(k^3)$. Instead, it would be proportional to the row and column dimension of the target block.

Our strategy to solve the problem facing matrix forward transformation is that, instead of preparing the admissible block form for each block b by a forward transformation in advance before the inverse, we will create it and update it level by level during the recursive inverse procedure. To solve the problem facing matrix backward transformation, when an auxiliary admissible block \mathbf{R}^b (this can be viewed as a counterpart of $\check{\mathbf{S}}_b^C$ used in a matrix–matrix multiplication) is generated during the

block-block multiplication, instead of splitting \mathbf{R}^b directly to its leaf blocks, we use $\mathbf{R}^b + \mathbf{G}^b$ as the real matrix block to perform next-level computation. The computation can be a $\mathbf{R}^b + \mathbf{G}^b$ based block matrix multiplication; it can also be a $\mathbf{R}^b + \mathbf{G}^b$ based inverse involved in the \mathbf{G}_{22}^{-1} part. For the former, we modify the block matrix multiplication algorithms. For the latter, we perform an instantaneous split procedure that has a linear complexity.

Along the above line of thought, we develop three new algorithms in the proposed inverse to render the total cost linear. The first algorithm is an instantaneous collect operation for generating the auxiliary admissible block form of \mathbf{G}^{-1} , \mathbf{X}_{12} , and \mathbf{X}_{21} . The second algorithm is a modified block matrix multiplication algorithm. The third one is an instantaneous split operation for computing the inverse of \mathbf{G}_{22} . To help better understand these three algorithms, the first algorithm can be viewed as the counterpart of the matrix forward multiplication. They fulfill the same task: when performing $\mathbf{G}^{b1} \times \mathbf{G}^{b2} \rightarrow \mathbf{G}^b$ or $\mathbf{G}^{b1} + \mathbf{G}^{b2} \rightarrow \mathbf{G}^b$, the auxiliary admissible block form of \mathbf{G}^{b1} and \mathbf{G}^{b2} should be ready so that each block matrix product or addition can be performed in constant complexity. The third algorithm can be viewed as the counterpart of matrix backward multiplication. Since the matrix forward and backward operations are modified, the block matrix multiplication should be modified correspondingly. That is the origin of the proposed second algorithm. In Section VI, we detail these three algorithms. Their corresponding pseudocodes are also given.

VI. ALGORITHMS IN THE PROPOSED INVERSE AND COMPLEXITY ANALYSIS

A. Instantaneous Collect Operation to Prepare the Auxiliary Admissible Block Form of \mathbf{G}^{-1} , \mathbf{X}_{12} , and \mathbf{X}_{21} in $O(N)$ Complexity

This operation can be viewed as the counterpart of the matrix forward transformation in [16], except that the collect operation is done instantaneously in the inverse procedure. As can be seen from (32), we need to perform a number of block matrix multiplications such as $\mathbf{G}_{21} \times \underline{\mathbf{G}}_{11} \rightarrow \mathbf{X}_{21}$, $\underline{\mathbf{G}}_{11} \times \mathbf{G}_{12} \rightarrow \mathbf{X}_{12}$, $\mathbf{X}_{21} \times \mathbf{G}_{12} \rightarrow \mathbf{G}_{22}$, etc. Here, the underlined \mathbf{G} is \mathbf{G}^{-1} . (Recall that, in the inverse procedure, after the computation at each level is done, \mathbf{G} is overwritten by its inverse.) Take $\mathbf{G}_{21} \times \underline{\mathbf{G}}_{11} \rightarrow \mathbf{X}_{21}$ as an example, to achieve the same complexity as that achieved in the linear-time matrix–matrix multiplication, we need to prepare for the auxiliary admissible block form of \mathbf{G}_{21} and $(\mathbf{G}^{-1})_{11}$ ($\underline{\mathbf{G}}_{11}$ is $(\mathbf{G}^{-1})_{11}$), respectively, denoting the two auxiliary admissible block forms by $\tilde{\mathbf{S}}_{\mathbf{G}_{21}}$ and $\tilde{\mathbf{S}}_{\mathbf{G}_{11}^{-1}}$. The former can still be prepared in advance, i.e., before the inverse procedure since \mathbf{G}_{21} is the original matrix. The latter, however, cannot be prepared in advance since $(\mathbf{G}^{-1})_{11}$ is updated level by level during the computation. To overcome this problem, our strategy is to generate $\tilde{\mathbf{S}}_{\mathbf{G}_{11}^{-1}}$ instantaneously through collect operation when $(\mathbf{G}^{-1})_{11}$ is computed. The procedure of a collect operation can be referred to [16, Algorithm 2].

As can be seen from (32), there are three matrices for which we need to collect their auxiliary admissible block form instantaneously during the inverse procedure: \mathbf{G}^{-1} (including \mathbf{G}_{11}^{-1} ,

\mathbf{G}_{22}^{-1} , and \mathbf{G}_{12}^{-1}), \mathbf{X}_{12} , and \mathbf{X}_{21} . Since these matrices are obtained by block matrix multiplications, the instantaneous collect operation can be performed in the level-by-level block matrix multiplication procedure that is given in Section VI-B. At each level, once the inadmissible block or a nonleaf block of the \mathbf{G}^{-1} , \mathbf{X}_{12} , or \mathbf{X}_{21} is computed, we perform a collect operation to obtain its auxiliary admissible block form. The algorithm for a collect operation used in the inverse is as follows:

```

Procedure CollectINV(b)
Form  $\tilde{\mathbf{S}}^b$  based on Algorithm 2 in [16]          (35)
If b is a nonleaf block
     $\tilde{\mathbf{S}}^b = \tilde{\mathbf{S}}^b + \mathbf{R}^b$ 

```

The collect operation is done level by level from bottom to top. The admissible form of each block at level l can be directly obtained from the four children blocks at level $l + 1$, instead of the blocks from level $l + 1$ all the way down to the leaf level. Therefore, each collect operation only costs $O(k_1^3)$ time. There are $O(N)$ blocks in \mathbf{G}^{-1} , \mathbf{X}_{12} , and \mathbf{X}_{21} . Each block is associated with one collect operation. Hence, the total complexity of performing the instantaneous collect operation for \mathbf{G}^{-1} , \mathbf{X}_{12} , and \mathbf{X}_{21} is linear.

For the original \mathbf{G}_{12} and original \mathbf{G}_{21} that are involved in the matrix multiplication, and the original \mathbf{G}_{22} involved in the matrix addition of (32), since they are from the original matrix, we can prepare an auxiliary admissible block form of \mathbf{G} in advance before the inverse procedure by using the matrix forward transformation [16, Algorithm 4], which has a linear complexity.

B. Modified Block Matrix Multiplication Algorithm of $O(N)$ Complexity for Inverse

Since neither matrix forward transformation, nor matrix backward transformation can be directly used in the proposed inverse, the algorithm for block matrix multiplications should also be modified. The matrix forward transformation is replaced by the instantaneous collect operation. Thus, when performing $\mathbf{G}^{b1} \times \mathbf{G}^{b2} \rightarrow \mathbf{G}^b$, we need to collect an admissible form for the target block b , $\tilde{\mathbf{S}}^b$, for the use of b -involved block matrix multiplication. In addition, for a nonleaf block b , the real matrix block stored in it could have a form of $\mathbf{G}^b + \mathbf{R}^b$ instead of only \mathbf{G}^b (this will become clear in Section VI-C). We cannot wait until the inverse is done to process \mathbf{R}^b by matrix backward transformation because \mathbf{R}^b is immediately involved in the next-level computation. Thus, we need to perform $(\mathbf{G}^{b1} + \mathbf{R}^{b1}) \times (\mathbf{G}^{b2} + \mathbf{R}^b) \rightarrow \mathbf{G}^b$ instead of $\mathbf{G}^{b1} \times \mathbf{G}^{b2} \rightarrow \mathbf{G}^b$ in the block matrix multiplication.

There are three basic block multiplication cases, i.e.: 1) admissible leaf as target; 2) inadmissible leaf as target; and 3) nonleaf as target. They correspond, respectively, to [16, Algorithms 7–9]. For the first case, we next show how to modify the block matrix multiplication algorithm to accommodate the need in the matrix inverse. Consider $\mathbf{G}^{b1} \times \mathbf{G}^{b2} \rightarrow \mathbf{G}^b$ with $b_1 = (t, s)$, $b_2 = (s, r)$, and $b = (t, r)$. The blocks b_1 , b_2 , and b can be in any form: an admissible form \mathbf{R} , an inadmissible

form \mathbf{F} , or a nonleaf form \mathbf{NL} . The possible b_1 and b_2 combinations that are involved in the block matrix multiplications are $\mathbf{R} - \mathbf{R}$, $\mathbf{NL} - \mathbf{NL}$, $\mathbf{F} - \mathbf{F}$, $\mathbf{F} - \mathbf{NL}$ (or $\mathbf{NL} - \mathbf{F}$), $\mathbf{R} - \mathbf{NL}$ (or $\mathbf{NL} - \mathbf{R}$), and $\mathbf{R} - \mathbf{F}$ (or $\mathbf{F} - \mathbf{R}$).

The algorithm for the modified block matrix multiplication with a target admissible leaf is developed as follows:

```

Procedure TargetAdmissibleINV( $b$ ) ( $b$  is an admissible leaf)
If  $b_1 - b_2$  combination is R-R, or F-F, or R-F
    Compute  $\mathbf{G}^{b_1} \times \mathbf{G}^{b_2} \rightarrow \mathbf{G}^b$  based on Algorithm 7
If  $b_1 - b_2$  combination is NL-NL
    Compute  $\mathbf{G}^{b_1} \times \mathbf{G}^{b_2} \rightarrow \mathbf{G}^b$  based on
        TargetAdmissibleINV( $b$ )
    Compute  $\mathbf{R}^{b_1} \times \mathbf{G}^{b_2} \rightarrow \mathbf{G}^b$ ,  $\mathbf{G}^{b_1} \times \mathbf{R}^{b_2} \rightarrow \mathbf{G}^b$ , and
         $\mathbf{R}^{b_1} \times \mathbf{R}^{b_2} \rightarrow \mathbf{G}^b$  based on Algorithm 7
If  $b_1 - b_2$  combination is R-NL or F-NL
    Compute  $\mathbf{G}^{b_1} \times \mathbf{G}^{b_2} \rightarrow \mathbf{G}^b$  based on Algorithm 7
    Compute  $\mathbf{G}^{b_1} \times \mathbf{R}^{b_2} \rightarrow \mathbf{G}^b$  based on Algorithm 7

```

As shown in the above, if the $b_1 - b_2$ combination is $\mathbf{R} - \mathbf{R}$, or $\mathbf{F} - \mathbf{F}$, or $\mathbf{R} - \mathbf{F}$ type, [16, Algorithm 7] can be directly used to compute the block matrix multiplication, the cost of which is at most $O(k_1^3)$. Once we meet the combination $\mathbf{NL} - \mathbf{NL}$, or $\mathbf{R} - \mathbf{NL}$, or $\mathbf{F} - \mathbf{NL}$, the block matrix multiplication has to be performed in a way that is different from that in [16, Algorithm 7]. If the $b_1 - b_2$ combination is $\mathbf{NL} - \mathbf{NL}$ type, \mathbf{R}^{b_1} and \mathbf{R}^{b_2} may be stored in b_1 and b_2 , respectively. Therefore, the real blocks that should be used are $\mathbf{G}^{b_1} + \mathbf{R}^{b_1}$ and $\mathbf{G}^{b_2} + \mathbf{R}^{b_2}$ instead of \mathbf{G}^{b_1} and \mathbf{G}^{b_2} . The block multiplication then becomes $(\mathbf{G}^{b_1} + \mathbf{R}^{b_1}) \times (\mathbf{G}^{b_2} + \mathbf{R}^{b_2}) \rightarrow \mathbf{G}^b$. To handle this multiplication, we separate it into two parts. One part is the original block multiplication $\mathbf{G}^{b_1} \times \mathbf{G}^{b_2} \rightarrow \mathbf{G}^b$, which belongs to the $\mathbf{NL} - \mathbf{NL} \rightarrow \mathbf{R}$ multiplication case. As shown in [16, Algorithm 7], the computation of $\mathbf{G}^{b_1} \times \mathbf{G}^{b_2} \rightarrow \mathbf{G}^b$ in this case involves recursive descendent-block matrix multiplications, each of which can be categorized into the basic block multiplication with an admissible leaf being a target and can be computed by recursively calling [16, Algorithm 7]. In the modified algorithm for inverse, we call the TargetAdmissible_{INV} recursively shown in (36). The other part is the three additional multiplications associated with $\mathbf{R}^{b_1(b_2)}$, i.e., $\mathbf{G}^{b_1} \times \mathbf{R}^{b_2} \rightarrow \mathbf{G}^b$, $\mathbf{R}^{b_1} \times \mathbf{G}^{b_2} \rightarrow \mathbf{G}^b$, and $\mathbf{R}^{b_1} \times \mathbf{R}^{b_2} \rightarrow \mathbf{G}^b$. They, in fact, belong to the multiplication cases of $\mathbf{NL} - \mathbf{R}$, $\mathbf{R} - \mathbf{NL}$, and $\mathbf{R} - \mathbf{R}$, respectively, with target being an admissible block. Each of these three cases can be performed in $O(k_1^3)$ complexity using [16, Algorithm 7].

If the $b_1 - b_2$ combination is $\mathbf{R} - \mathbf{NL}$ or $\mathbf{F} - \mathbf{NL}$ type, similar to $\mathbf{NL} - \mathbf{NL}$ type, we separate the computation to $\mathbf{G}^{b_1} \times \mathbf{G}^{b_2} \rightarrow \mathbf{G}^b$ and $\mathbf{G}^{b_1} \times \mathbf{R}^{b_2} \rightarrow \mathbf{G}^b$. The latter is a case of $\mathbf{R} - \mathbf{R}$ or $\mathbf{F} - \mathbf{R}$ multiplication with the target block being an admissible block. It again can be performed in $O(k_1^3)$ complexity based on [16, Algorithm 7].

Since \mathbf{G}^b itself is an admissible block, we do not need to perform a collect operation to prepare its auxiliary admissible block form $\tilde{\mathbf{S}}^b$.

Consider the block matrix multiplication with an inadmissible block being a target block. We develop the following

pseudocode:

```

Procedure TargetDenseINV( $b$ ) ( $b$  is an admissible leaf)
If  $b_1 - b_2$  combination is F-F, or R-F, or R-R
    Compute  $\mathbf{G}^{b_1} \times \mathbf{G}^{b_2} \rightarrow \mathbf{G}^b$  based on Algorithm 8
If  $b_1 - b_2$  combination is R-NL or F-NL
    Compute  $\mathbf{G}^{b_1} \times \mathbf{G}^{b_2} \rightarrow \mathbf{G}^b$  based on TargetDenseINV( $b$ )
    Compute  $\mathbf{G}^{b_1} \times \mathbf{R}^{b_2} \rightarrow \mathbf{G}^b$  based on Algorithm 8
CollectINV( $b$ )

```

As can be seen from the above, if $b_1 - b_2$ combination is F-NL or R-NL, we separate the computation to $\mathbf{G}^{b_1} \times \mathbf{G}^{b_2} \rightarrow \mathbf{G}^b$ and $\mathbf{G}^{b_1} \times \mathbf{R}^{b_2} \rightarrow \mathbf{G}^b$. The latter one can be directly handled by [16, Algorithm 8]. $\mathbf{G}^{b_1} \times \mathbf{G}^{b_2} \rightarrow \mathbf{G}^b$ involves recursive descendent-block matrix multiplications with inadmissible targets, each of which can be computed by recursively calling (37) instead of [16, Algorithm 8]. In addition, since the target is a full matrix block, for efficient computation, during the recursive computation, we do not perform the collect operation on the block intermediate results, but do the collect operation on the target block when the block matrix multiplication is done, as can be seen from (37). All the other $b_1 - b_2$ combinations in (37) can be directly computed based on [16, Algorithm 8]. In (37), each block matrix multiplication costs $O(k_1^3)$ time. After the full matrix target block is computed, we compute its $\tilde{\mathbf{S}}^b$ form by performing a collect operation, the cost of which is at most $O(n_{\min}^2 k)$.

The modification to the third block multiplication case, i.e., the case with nonleaf as a target, can be derived in a similar way. Basically, the computation of $(\mathbf{G}^{b_1} + \mathbf{R}^{b_1}) \times (\mathbf{G}^{b_2} + \mathbf{R}^{b_2}) \rightarrow \mathbf{G}^b$ is separated into two parts. One part is the original $\mathbf{G}^{b_1} \times \mathbf{G}^{b_2} \rightarrow \mathbf{G}^b$. The other part is \mathbf{R} -based computation. The second part involves three multiplications, each of which can be categorized as one case of the block multiplications that are handled by [16, Algorithms 7–9]. The procedure for this basic multiplication case is shown as follows:

```

Procedure TargetNonleafINV( $b$ ) ( $b$  is a nonleaf)
If  $b_1 - b_2$  combination is R-R or R-F
    Compute  $\mathbf{G}^{b_1} \times \mathbf{G}^{b_2} \rightarrow \mathbf{R}^b$  based on (36)
     $\tilde{\mathbf{S}} = \tilde{\mathbf{S}}^b + \mathbf{R}^b$ 
else
    If  $b_1 - b_2$  combination is F-F
        Compute  $\mathbf{G}^{b_1} \times \mathbf{G}^{b_2} \rightarrow \mathbf{G}^b$  based on TargetDenseINV( $b$ )
    If  $b_1 - b_2$  combination is R-NL
        Compute  $\mathbf{G}^{b_1} \times \mathbf{R}^{b_2} \rightarrow \mathbf{R}^b$  based on (36)
        Compute  $\mathbf{G}^{b_1} \times \mathbf{G}^{b_2} \rightarrow \mathbf{G}^b$  based on TargetDenseINV( $b$ )
    If  $b_1 - b_2$  combination is NL-NL
        Compute  $\mathbf{R}^{b_1} \times \mathbf{G}^{b_2} \rightarrow \mathbf{G}^b$ ,  $\mathbf{G}^{b_1} \times \mathbf{R}^{b_2} \rightarrow \mathbf{G}^b$  based on
            TargetNonleafINV( $b$ ) and  $\mathbf{R}^{b_1} \times \mathbf{R}^{b_2} \rightarrow \mathbf{R}^b$  based on (36)
        Compute  $\mathbf{G}^{b_1} \times \mathbf{G}^{b_2} \rightarrow \mathbf{G}^b$  based on TargetNonleafINV( $b$ )
CollectINV( $b$ )

```

The instantaneous collect operation for each target block is done during the block matrix multiplication.

In the modified block matrix multiplication derived in this work, we employ (36)–(38) to handle a block matrix multiplication with the target block being any form. The computation for each $b_1 - b_2$ multiplication case performed by calling (36)–(38) has the same order of complexity as the corresponding multiplication case handled by [16, Algorithms 7–9]. As proven in [16], for matrix–matrix multiplication, the three basic multiplication algorithms (admissible leaf as target, inadmissible leaf as target, and nonleaf as target) are called no more than $O(3C_{sp}^2 N)$ times. The same is true in matrix inverse since it shares the same number of block multiplications with a matrix–matrix product, as analyzed in Section V-C. The computation involved in each call costs at most $O(k_1^3)$ operations. This includes the cost of the additional multiplications associated with \mathbf{R}^b . The total cost of the modified block matrix multiplications in the proposed inverse is, hence, $O(C_{sp}^2 k_1^3 N)$, which is linear. The cost of the instantaneous collect operation has already been counted in Section VI-A.

C. $O(N)$ Instantaneous Split Operation for Computing \mathbf{G}_{22}^{-1}

As mentioned before, a block multiplication can generate an auxiliary block \mathbf{R}^b for a nonleaf block \mathbf{G}^b , and hence, $\mathbf{R}^b + \mathbf{G}^b$ is used as the real matrix for b . If \mathbf{G}_{22} is a nonleaf block, to compute its inverse, we need to compute $(\mathbf{G}_{22} + \mathbf{R})^{-1}$ instead of \mathbf{G}_{22}^{-1} . Unlike the \mathbf{R} -associated computation in a block matrix multiplication, it is difficult to separate $(\mathbf{G}_{22} + \mathbf{R})^{-1}$ into \mathbf{G}_{22} -associated and \mathbf{R} -associated computation. In order to compute $(\mathbf{G}_{22} + \mathbf{R})^{-1}$ efficiently, based on a Split operation [16, Algorithm 1], we first obtain $\mathbf{G}_{22} + \mathbf{R}$ by splitting \mathbf{R} to \mathbf{G}_{22} 's children blocks. The pseudocode of this procedure is shown as follows:

Procedure $\text{Split}_{\text{INV}}(b, \mathbf{R}^b)$ (b is a 22-position nonleaf block)

Apply Algorithm 1 to \mathbf{R}^b to form four children $\tilde{\mathbf{R}}^{bij}$

for $i = 1, 2$ and $j = 1, 2$

if b_{ij} is an admissible block

$$\mathbf{S}^{bij} = \mathbf{S}^{bij} + \tilde{\mathbf{R}}^{bij} \quad (\text{update the coupling matrix})$$

else

if b_{ij} is a full matrix block

$$\mathbf{F}^{bij} = \mathbf{F}^{bij} + \mathbf{V}^{ti} \tilde{\mathbf{R}}^{bij} \mathbf{V}^{sjT} \quad (\text{update the full matrix})$$

if b_{ij} is a nonleaf block

$$\mathbf{R}^{bij} = \mathbf{R}^{bij} + \tilde{\mathbf{R}}^{bij} \quad (\text{update R block at children level}) \quad (39)$$

$$\tilde{\mathbf{S}}^{bij} = \tilde{\mathbf{S}}^{bij} + \tilde{\mathbf{R}}^{bij} \quad (\text{update the collected admissible block})$$

Clear \mathbf{R}^b

Based on (39), \mathbf{R} is superposed with \mathbf{G}_{22} . We can then compute \mathbf{G}_{22}^{-1} . Since the inverse procedure is recursive, in order to compute the inverse of the nonleaf \mathbf{G}_{22} , we have to first compute the inverse of \mathbf{G}_{22} 's 11 child block and 22 child block. If 11 and 22 blocks are both nonleaf blocks, in order to compute their inverses, we again need to split the \mathbf{R} blocks in the 11

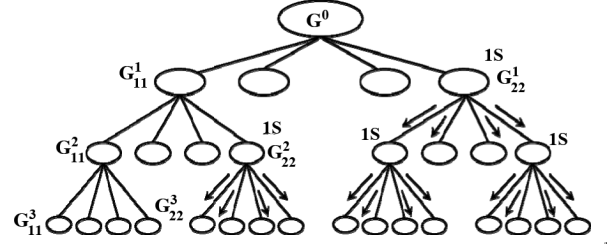


Fig. 6. Illustration of the instantaneous split operation for computing \mathbf{G}_{22}^{-1} .

and 22 blocks, respectively, to their children. This process continues until 11 and 22 blocks become full matrices, the inverse of which can be directly computed. The aforementioned procedure is illustrated in Fig. 6, and its corresponding pseudocode is shown as follows:

Procedure $\text{H}^2\text{-inverse}_{22}(\mathbf{G}, \mathbf{X})$ (\mathbf{G} is a 22-position nonleaf block)

if matrix \mathbf{G} is a nonleaf matrix block

Split_{INV}(\mathbf{G}, \mathbf{R})

$\text{H}^2\text{-inverse}_{22}(\mathbf{G}_{11}, \mathbf{X}_{11})$

$$\mathbf{G}_{21} \times \mathbf{G}_{11} \rightarrow \mathbf{X}_{21}, \mathbf{G}_{11} \times \mathbf{G}_{12} \rightarrow \mathbf{X}_{12},$$

$$\mathbf{G}_{22} + (-\mathbf{X}_{21} \times \mathbf{G}_{12}) \rightarrow \mathbf{G}_{22},$$

$\text{H}^2\text{-inverse}_{22}(\mathbf{G}_{22}, \mathbf{X}_{22})$

$$-\mathbf{G}_{22} \times \mathbf{X}_{21} \rightarrow \mathbf{G}_{21}, -\mathbf{X}_{12} \times \mathbf{G}_{22} \rightarrow \mathbf{G}_{12},$$

$$\mathbf{G}_{11} + (-\mathbf{G}_{12} \times \mathbf{X}_{21}) \rightarrow \mathbf{G}_{11},$$

else

DirectInverse(\mathbf{G}) (normal full matrix inverse)

(40)

As can be seen from Fig. 6 and (40), the nonleaf \mathbf{G}_{22} blocks and all their descendant nonleaf 11 and 22 blocks each is associated with one “Split” operation denoted by “1S.”

The cost of each Split operation from the parent level to the children that is one level down is at most $O(k_1^3)$ [16]. This operation is only done for the nonleaf \mathbf{G}_{22}^l at each level l and its descendant nonleaf 11 and 22 blocks. Therefore, the processed blocks only cover a part of the entire \mathcal{H}^2 partition, as can be seen from Fig. 6. Since the total number of blocks is $O(C_{sp} N)$ and each Split operation costs $O(k_1^3)$ time, the complexity of the instantaneous split in the inverse procedure is bounded by $O(C_{sp} k_1^3 N)$, which is linear.

D. $O(N)$ Backward Transformation After the Inverse Procedure

After the inverse procedure is done, \mathbf{R}^b may be stored for a nonleaf block b in a block cluster tree. For an \mathcal{H}^2 matrix, all the matrix elements are actually stored in leaf blocks. Therefore, \mathbf{R}^b stored in each nonleaf block should be distributed back to leaf blocks to obtain a final \mathcal{H}^2 matrix. This can be achieved by the matrix backward transformation after the inverse procedure, which has a linear complexity.

VII. ACCURACY ANALYSIS

There exist three error sources in the proposed direct solver, i.e.: 1) \mathcal{H}^2 -based representation of the original matrix; 2) or-

thogonalization; and 3) \mathcal{H}^2 -based inverse. Next, we analyze the three errors one by one.

First, the \mathcal{H}^2 -based representation of the dense matrix resulting from an IE-based analysis of capacitance extraction problem is error bounded, as shown in Section II. Exponential convergence with respect to the number of interpolation points, p , can be achieved irrespective of the problem size.

Second, the orthogonalization error can be minimized to zero. In Section IV, orthogonal bases $\tilde{\mathbf{V}}^t$ are constructed. The best approximation of a general \mathbf{V}^t in the space $\tilde{\mathbf{V}}^t$ is given by $\tilde{\mathbf{V}}^t(\tilde{\mathbf{V}}^t)^\top \mathbf{V}^t$. The error of this approximation is

$$\|\mathbf{V}^t - \tilde{\mathbf{V}}^t(\tilde{\mathbf{V}}^t)^\top \mathbf{V}^t\|_2^2 = \lambda_{k^t+1} \quad (41)$$

where λ_{k^t+1} is the $(k^t + 1)$ th eigenvalue of $\mathbf{V}^{t\top} \mathbf{V}^t$, in which k^t is the rank of cluster basis $\tilde{\mathbf{V}}^t$. Clearly, if k^t is chosen the same as the rank of \mathbf{V}^t , the error of (41) is zero. Therefore, $\tilde{\mathbf{V}}^t \tilde{\mathbf{V}}^{t\top} \mathbf{G} \tilde{\mathbf{V}}^s \tilde{\mathbf{V}}^{s\top}$ is the best approximation of a matrix block $\mathbf{G}^{t,s}$ in the bases $\tilde{\mathbf{V}}^t$ and $\tilde{\mathbf{V}}^s$.

Third, the inverse has a controlled accuracy. If one agrees with the fact that the linear-time matrix–matrix multiplication developed in [16] has a controlled accuracy, the same is true for the proposed inverse since the inverse procedure is essentially a full matrix inverse at leaf level, and a level-by-level block matrix multiplication procedure at nonleaf levels. The new instantaneous collect algorithm added for inverse has the same accuracy as the matrix forward transformation since the basic operations are the same. Similarly, the new instantaneous split operation has the same accuracy as the matrix backward transformation in the linear-time matrix–matrix multiplication algorithm. The modified block matrix multiplication algorithm has the same accuracy as the original one since although three additional multiplications are added; they are done with the same accuracy. In addition, it is worth mentioning that no pivoting is needed in the proposed inverse since capacitance matrix is a diagonally dominant matrix.

The inverse accuracy can also be analyzed from another perspective. The inverse procedure is essentially a number of block matrix multiplications. The multiplication is performed by a formatted multiplication in which the \mathcal{H}^2 tree of \mathbf{G}^{-1} is represented by the \mathcal{H}^2 tree of \mathbf{G} . In addition, the same cluster basis used for \mathbf{G} is used for \mathbf{G}^{-1} . Both have been theoretically proven to be true in Section II-B.

From the aforementioned three facts, the accuracy of the proposed direct solver is well controlled.

VIII. NUMERICAL RESULTS

A number of examples were simulated to validate the accuracy and demonstrate the linear complexity of the proposed direct IE solver. For all these simulations, a Dell 1950 Server was used, except for the comparison with HiCap [20], where a computer having a 1593-MHz SPARC v9 processor was used since HiCap available in the public domain can only be run on a Sun SPARC platform.

There are only three simulation parameters: η , leafsize n_{\min} , and p to choose in the proposed method. From (16), it can be seen that the smaller the η and the larger the p , the better the accuracy. For static problems, $1 \leq \eta \leq 2$ is generally sufficient

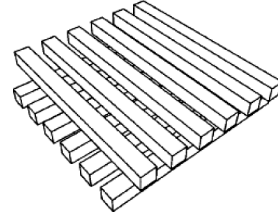


Fig. 7. $m \times m$ crossing bus structure.

for achieving good accuracy. With η chosen, based on accuracy requirements, one can choose p accordingly. The leafsize, n_{\min} , can be chosen based on $n_{\min} \geq 0.5 p^d$. This can help make the \mathcal{H}^2 -approximation more efficient in both memory and CPU time.

The first example is an $m \times m$ crossing bus structure embedded in free space [3], as shown in Fig. 7, where m is from 4 to 16. The dimension of each bus is scaled to $1 \times 1 \times (2m+1) m^3$. The spacing between buses in the same layer is 1 m, and the distance between the two bus layers is 1 m. Although meter is not a realistic on-chip length unit, it should be noticed that capacitances are scalable with respect to the length unit.

We first compared the performance of the proposed direct solver with FastCap 2.0. The discretization in FastCap 2.0 resulted in 2736–38 592 unknowns for the extraction of the $m \times m$ bus from $m = 4$ to $m = 16$. A similar number of unknowns were also generated in the proposed solver for a fair comparison. The convergence tolerance was set to 1% when using FastCap. The simulation parameters in the proposed solver were chosen as $n_{\min} = 10$ and $\eta = 1.6$. The number of interpolation points p was determined by a function $p = a + b(l - l)$, with $a = 2$, $b = 1$, and L being the maximum number of tree level, and l being tree level. Such a choice of p reduces the \mathcal{H}^2 -approximation error without affecting the linear cost [18].

In Fig. 8, we plot the original matrix error, which is the error of the \mathcal{H}^2 -based representation of the original matrix \mathbf{G} , as well as the error of the capacitance matrix with respect to the number of unknowns. The original matrix error is measured by $\|\mathbf{G} - \tilde{\mathbf{G}}\|_F / \|\mathbf{G}\|_F$, where $\tilde{\mathbf{G}}$ is the \mathcal{H}^2 -matrix representation shown in (10), and $\|\cdot\|_F$ is the Frobenius norm; the capacitance error is measured by $\|\mathbf{C} - \mathbf{C}'\|_F / \|\mathbf{C}\|_F$, where \mathbf{C} is the capacitance matrix obtained from a full-matrix-based direct solver, and \mathbf{C}' is that generated by the proposed solver. As can be seen clearly from Fig. 8, excellent accuracy of the proposed direct solver can be observed in both $\tilde{\mathbf{G}}$ and capacitance matrix \mathbf{C}' . In addition, the error of $\tilde{\mathbf{G}}$ is shown to reduce with the number of unknowns. This is because of increased p with respect to tree level, and hence, increased accuracy, as can be seen from (16). In addition, we are able to keep the accuracy of the capacitance to the same order in the entire range.

With the accuracy of the proposed direct solver validated, in Fig. 9, we plot the total CPU time and memory consumption of the proposed direct solver for the $m \times m$ bus structure in free space. As can be clearly seen, both time and memory complexity of the proposed solver are linear. In addition, in Fig. 9, we plot the CPU time and memory cost of FastCap2.0. It is clear that the proposed direct solver outperforms FastCap2.0. In addition, FastCap2.0 does not exhibit a linear scaling with respect to the

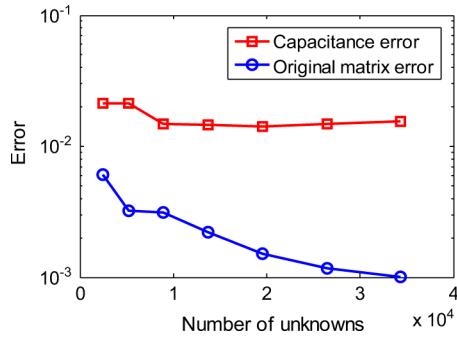


Fig. 8. Original matrix error and capacitance error of the proposed solver with respect to N for the free-space case.

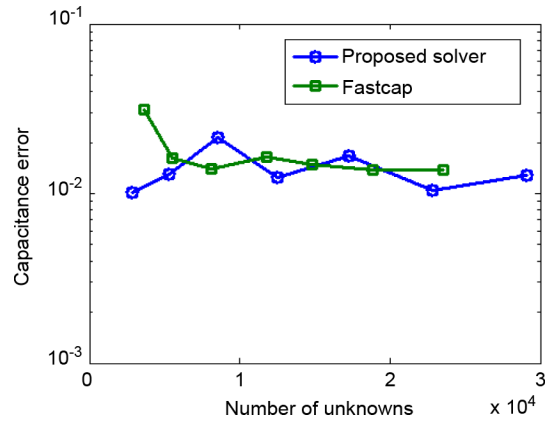


Fig. 10. Capacitance error of the proposed solver and that of FastCap2.0 for the nonuniform dielectric case.

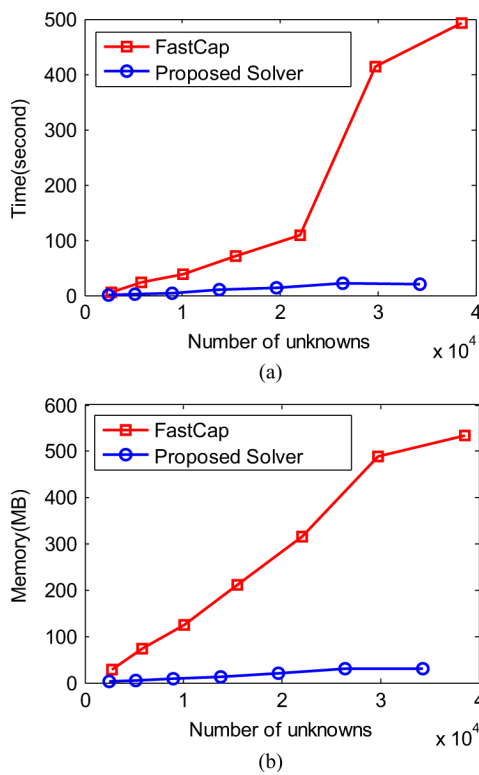


Fig. 9. Comparison of time and memory complexity in simulating the bus structure in free space. (a) Time complexity. (b) Memory complexity.

number of unknowns although it performs matrix–vector multiplication in linear complexity. This could be attributed to the increased number of iterations when the number of unknowns increases.

Next, we simulated the same bus structure embedded in nonuniform dielectrics. The dielectric surrounding the upper layer conductors has relative permittivity of 3.9, and that surrounding the lower layer has relative permittivity 7.5. Each bus is again scaled to $1 \times 1 \times (2m + 1) m^3$. The distance between buses in the same layer is 1 m, and the distance between the two bus layers is 2 m. The discretization in FastCap 2.0 resulted in 3636 to 23 552 unknowns for the extraction of the $m \times m$ bus from $m = 4$ to $m = 16$. A similar number of unknowns were generated in the proposed solver.

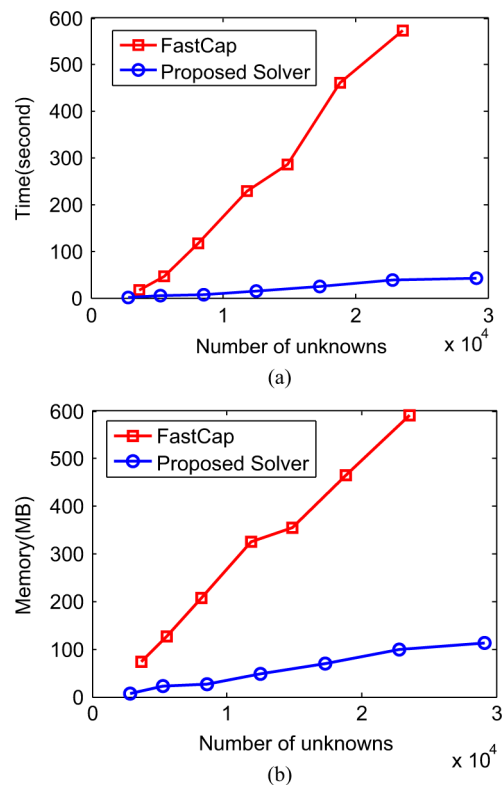


Fig. 11. Comparison of time and memory complexity in simulating the bus structure embedded in multiple dielectrics. (a) Time complexity. (b) Memory complexity.

The simulation parameters of the proposed solver can be chosen to achieve a various level of accuracy. For a fair comparison with FastCap2.0, we chose the simulation parameters in such a way that the proposed solver and FastCap2.0 produced similar accuracy in capacitance, as shown in Fig. 10, where the reference capacitance matrix \mathbf{C} for both solvers was chosen as that generated by a full-matrix-based direct calculation. The resultant simulation parameters were leafsize $n_{\min} = 10$, $a = 2$, and $b = 1$. We then compared the time and memory performance of the two solvers. In Fig. 11, we plot the total CPU time and memory consumption of the proposed direct solver for the $m \times m$ bus structure in nonuniform dielectrics,

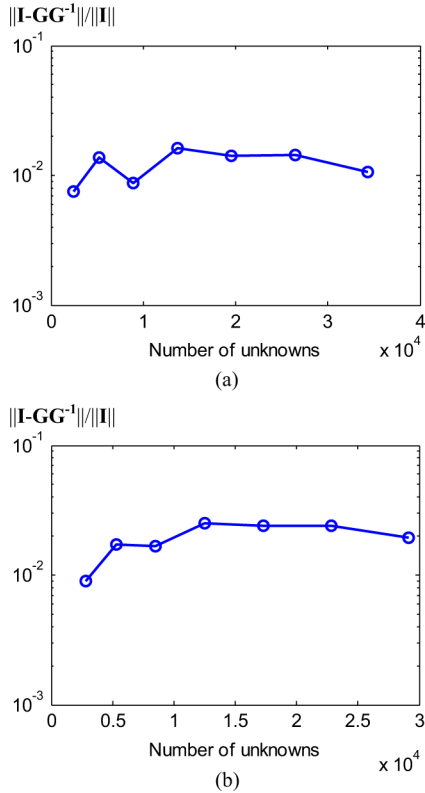


Fig. 12. Inverse error of the proposed direct solver. (a) Free-space case. (b) Nonuniform case.

and compare the performance with FastCap2.0. Once again, the linear complexity of the proposed direct IE solver can be clearly seen in both CPU time and memory consumption. It is also worth mentioning that the proposed solver used double precision to carry out the computation. If single precision was used, more CPU time and memory usage can be saved. In addition, we notice that for capacitance extraction, single precision is generally sufficient to achieve good accuracy.

Since capacitance extraction does not involve all the columns of \mathbf{G}^{-1} , to assess the accuracy of the entire inverse, in Fig. 12, we plot the inverse error versus unknown number for both free-space and nonuniform dielectric cases. Good accuracy is observed in the entire range. The inverse error is assessed by $\|\mathbf{I} - \mathbf{G}\mathbf{G}^{-1}\|_F / \|\mathbf{I}\|_F$. The simulation parameters were $n_{\min} = 10$ and $\eta = 1.6$. The number of interpolation points, p , was 2.

Next, we compared the performance of the proposed direct solver with HiCap downloaded from [20]. This version of HiCap is for simulating free-space examples, and allows for at most a 20×20 bus. We hence compared the performance of simulating the free-space $m \times m$ bus from $m = 4$ to $m = 20$. The number of unknowns used in HiCap was from 1104 to 20880. A similar number of unknowns were generated in the proposed direct solver for a fair comparison. The number of unknowns used in the proposed direct solver was from 1216 to 26560. The simulation parameters in the proposed solver were chosen as leafsize = 8, $\eta = 1.2$, and $p = 1$. Fig. 13 shows the inverse error in the entire range. Good accuracy can be observed. In Fig. 14(a)–(c), we plot the total CPU time, memory

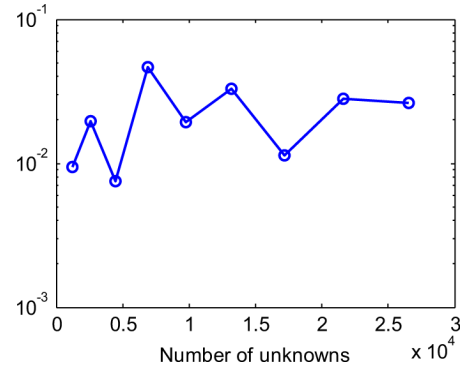


Fig. 13. Inverse error $\|\mathbf{I} - \mathbf{G}\mathbf{G}^{-1}\| / \|\mathbf{I}\|$ versus N .

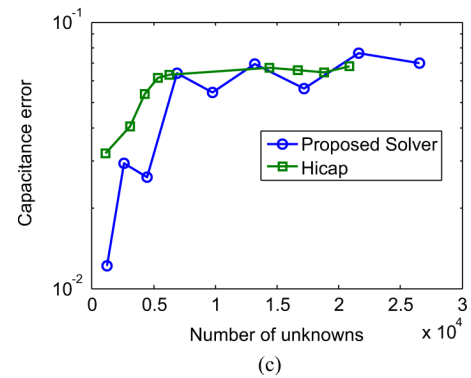
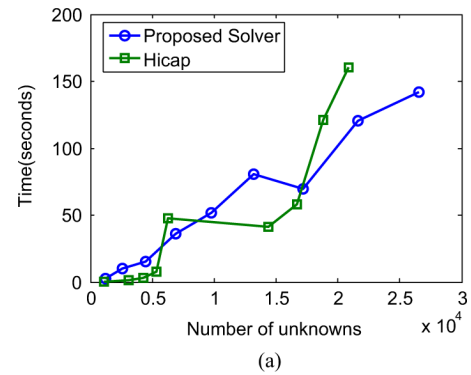


Fig. 14. Comparison with HiCap in simulating an $m \times m$ bus with m being from 4 to 20. (a) CPU time. (b) Memory. (c) Capacitance error.

consumption, and capacitance error of the proposed solver and those of HiCap. The capacitance error was measured by $\|\mathbf{C} - \mathbf{C}'\|_F / \|\mathbf{C}\|_F$, where the reference \mathbf{C} was obtained from a full-matrix-based direct solver. The simulation parameters of the

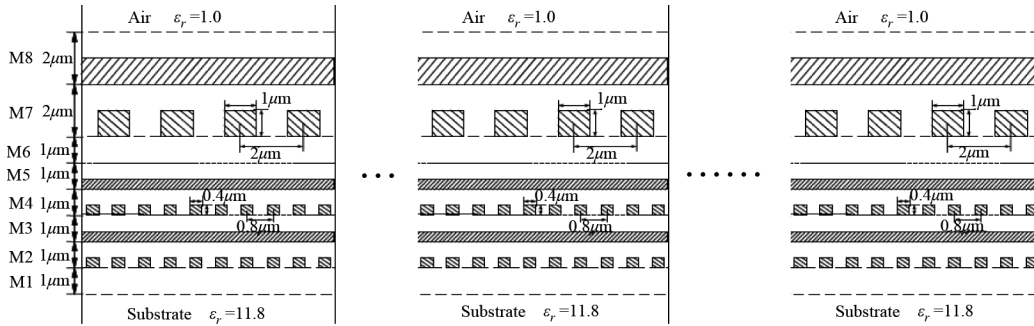


Fig. 15. Large-scale 3-D M1–M8 on-chip interconnect embedded in inhomogeneous media.

proposed solver were chosen such that both solvers yielded a similar level of accuracy, as can be seen from Fig. 14(c). From Figs. 14(a) and (b), it can be seen that HiCap starts to become more expensive in both CPU time and memory consumption when problem size becomes large. In addition, the accuracy of the proposed solver is shown to be better than HiCap on average. Considering the fact that HiCap only solved the matrix for 4–20 right-hand sides in simulating this bus structure, whereas the proposed solver computed the entire inverse, the performance of the proposed direct solver is satisfactory.

To test the performance of the proposed direct solver in simulating very large examples, we simulated a multilayer 3-D on-chip interconnect structure [3] shown in Fig. 15. We also compared the performance of the proposed direct solver with a HiCap-based solver in this simulation. The relative permittivity of the interconnect structure is 3.9 in M1, 2.5 from M2 to M6, and 7.0 from M7 to M8. The structure involves 48 conductors, the discretization of which results in 25 556 unknowns. To test the large-scale modeling capability of the proposed solver, the 48-conductor structure was duplicated horizontally, resulting in 72, 96, 120, 144, 192, 240, 288, and 336 conductors, the discretization of which leads to more than one million unknowns including both conducting-surface unknowns and dielectric-interface unknowns.

The simulation parameters in the proposed solver were chosen as leafsize = 10, $\eta = 1$, and $p = 1$. Since it is not feasible to assess the error of \mathcal{H}^2 -matrix-based representation based on $\|\mathbf{G} - \tilde{\mathbf{G}}\|_F / \|\mathbf{G}\|_F$ due to the need of storing the original dense matrix \mathbf{G} , we plot the maximal admissible block error of the proposed solver in Fig. 16(a). The maximal admissible block error is defined as

$$\max \left\{ \frac{\|\mathbf{G}^{(t,s)} - \tilde{\mathbf{G}}^{(t,s)}\|}{\|\mathbf{G}^{(t,s)}\|} \right\}$$

which constitutes an upper bound of the entire matrix error $\|\mathbf{G} - \tilde{\mathbf{G}}\|_F / \|\mathbf{G}\|_F$. As can be seen from Fig. 16(a), less than 2% error is observed in the entire range from 25 556 unknowns to 1 047 236 unknowns. In Fig. 16(b), we plot the inverse time and the total CPU time of the proposed direct solver with respect to the number of unknowns. Clearly, a linear complexity can be observed. The total CPU time of the proposed direct solver includes orthogonalization time, inverse time, and matrix–vector multiplication time for computing unknown charge vector and capacitances. For comparison, the solution time of a

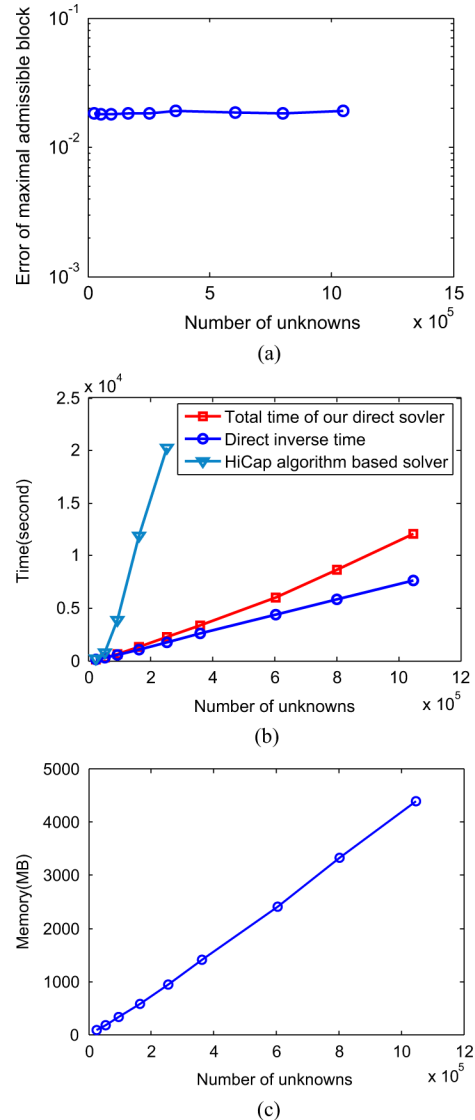


Fig. 16. Simulation of a large-scale 3-D M1–M8 on-chip interconnect. (a) Error of maximal admissible block. (b) CPU time. (c) Memory.

HiCap-based solver is also plotted in Fig. 16(b). Since HiCap for inhomogeneous dielectrics is not available in public domain, we generated the HiCap time in the following way to make the comparison as fair as possible. We first constructed an \mathcal{H}^2 -based representation of \mathbf{G} with $p = 1$ since the center-point based scheme in HiCap can be viewed as a rank 1 scheme. We then performed

TABLE I
SOLUTION ERROR VERSUS THE UNKNOWN NUMBER

Num. of Unknowns	Solution Error (%)
25,556	3.33
53,400	5.01
94,752	5.06
164,672	7.26
253,792	6.63
362,122	5.28
605,472	5.59
802,272	6.23
1,047,236	5.98

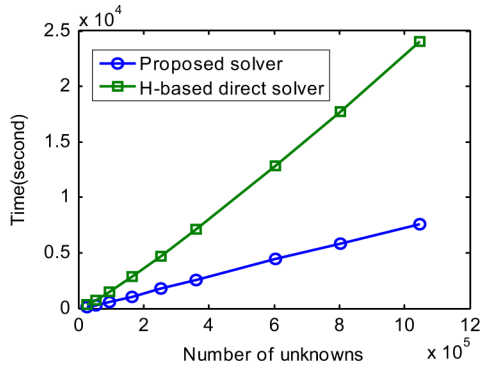


Fig. 17. Inverse time comparison between the proposed solver and an \mathcal{H} -based direct solver.

a matrix–vector multiplication based on the \mathcal{H}^2 -based representation, which has a similar CPU time as that reported in [3] if run on the same computer platform. With the CPU time per matrix–vector multiplication matched, we chose the same number of iterations as reported in [3] to generate the CPU time required by a HiCap algorithm based solver.

As can be seen from Fig. 16, the advantage of the proposed direct solver is clearly demonstrated even though a HiCap-based solver only calculated the results for m right-hand sides with m being the number of conductors, whereas the proposed solver obtained the entire inverse, i.e., the results for N right-hand sides. In Fig. 16(c), we plot the memory complexity of the proposed solver, which again demonstrates a linear complexity.

Since we need to use the capacitance \mathbf{C} generated from a full-matrix based direct computation to assess the accuracy of the capacitance \mathbf{C}' extracted by the proposed solver, and \mathbf{C} is not available within feasible computational resources for this large example, we tested the solution error of the proposed solver, which is defined as $\|\mathbf{G}q - v\| / \|v\|$. Table I shows the solution error in the entire range. Good accuracy is observed even with $p = 1$.

The best complexity reported for the IE-based direct solver is $O(N \log^\alpha N)$ [10], [24]–[26], which is higher than $O(N)$. Next, we compare the proposed linear direct solver with an $O(N \log^2 N)$ complexity \mathcal{H} -based direct solver [10]–[12], [26]. In order to have a fair comparison, we employ the same matrix partition to form an \mathcal{H} -based matrix. In addition, the interpolation-based rank used in the \mathcal{H} -based block is the same as that in the \mathcal{H}^2 -based block. The direct inverse of such an \mathcal{H} -based matrix can be developed based on the direct inverse algorithm

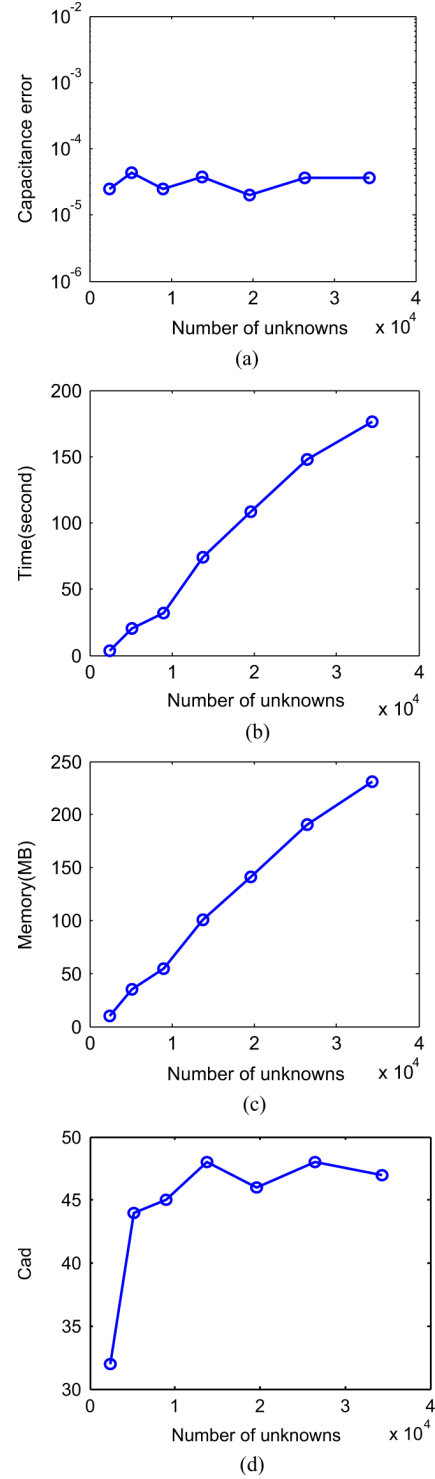


Fig. 18. Performance of the proposed solver in achieving a higher order of accuracy. (a) Capacitance error. (b) Time complexity. (c) Memory. (d) Sparsity constant.

given in [26], which has an $O(N \log^2 N)$ complexity. Fig. 17 compares the inverse time of the proposed solver with that of the \mathcal{H} -based direct solver. Clearly, the proposed solver is shown to be much faster than the \mathcal{H} -based direct solver. When the number of unknowns is larger, the advantage of the proposed solver will be even more obvious.

In the last example, we tested the capability of the proposed solver in achieving a higher order of accuracy. We set the required level of accuracy measured by capacitance error to be 10^{-5} . The structure was the 3-D bus shown in Fig. 7. The simulation parameters of the proposed solver were chosen as $n_{\min} = 32$, $\eta = 1$, $a = 3$, and $b = 1$ to satisfy the required accuracy. As shown in Fig. 18(a), the required accuracy is achieved across the entire range of unknowns without sacrificing the linear complexity in CPU time and memory consumption. This is clearly demonstrated in Fig. 18(b) and (c). We tried to use either FastCap or HiCap that can be accessed from the public domain to produce 10^{-5} accuracy in capacitances so that we can compare the performance for the same accuracy. However, when we decreased the convergence tolerance or increased the expansion order to a certain extent, the accuracy of the two solvers became saturated. They failed to produce a 10^{-5} level of accuracy in capacitances. In Fig. 18(d), we plot C_{ad} , the maximal number of admissible blocks formed by a cluster, which is a good measurement of C_{sp} . The C_{ad} is almost a constant in the entire range of unknowns, as can be seen from Fig. 18(d).

IX. CONCLUSION

In this paper, we have shown that the dense matrix arising from the IE-based analysis of capacitance problems can be represented by an \mathcal{H}^2 matrix with error well controlled. In addition, we have theoretically proven that the inverse of this dense matrix, also, has an \mathcal{H}^2 representation. More important, the same block cluster tree and cluster bases constructed from the original dense matrix can be used for the \mathcal{H}^2 representation of its inverse. Based on this finding, we develop a direct inverse of linear complexity for large-scale capacitance extraction involving arbitrary inhomogeneity and arbitrary geometry. To help better convey the idea of the proposed linear-time inverse, we use an analogy between a matrix–matrix product and a matrix inverse to present the proposed algorithm. We show that these two matrix operations share the same number of block matrix multiplications. However, in the matrix inversion procedure, the matrix blocks used for computation are kept updated level by level. In contrast, in a matrix–matrix multiplication, the matrix blocks used for computation at each level are always from the original matrix. They are never updated. This difference makes it not feasible to achieve a linear complexity in inverse by directly using the linear-time matrix–matrix multiplication algorithm. We then present the proposed algorithms that achieve a linear complexity in inverse. Both theoretical analysis and numerical results have demonstrated the accuracy and linear complexity of the proposed direct IE solver. In addition, the proposed direct solver is shown to outperform existing iterative IE solvers of linear complexity. The proposed solver is kernel independent in the sense that it does not rely on an analytical expansion of kernels, and the underlying fast techniques are algebraic methods that are not kernel specific. Moreover, it is applicable to arbitrary inhomogeneity and arbitrary structures.

In this paper, we demonstrate that it is feasible to obtain an inverse of a dense matrix in linear time and memory consumption with controllable accuracy. Inverse is a fundamental building

block in computation. The significance of the proposed work goes beyond just capacitance extraction.

ACKNOWLEDGMENT

The authors would like to thank Prof. C.-K. Koh, Purdue University, West Lafayette, IN, for valuable suggestions to this work. The authors also appreciate the interaction with Prof. J. White, Massachusetts Institute of Technology, Boston, on FastCap.

REFERENCES

- [1] K. Nabors and J. White, "FastCap: A multipole accelerated 3-D capacitance extraction program," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, 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. Comput.-Aided Design Integr. Circuits Syst.*, 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 *DAC*, 2004, pp. 788–793.
- [4] S. Kapur and D. E. Long, "IES³: A fast integral equation solver for efficient 3-dimensional extraction," in *Proc. ICCAD*, 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. ICCAD*, 1994, pp. 268–271.
- [6] D. Gope, I. Chowdhury, and V. Jandhyala, "DiMES: Multilevel fast direct solver based on multipole expansions for parasitic extraction of massively coupled 3-D microelectronic structures," in *DAC*, 2005, pp. 159–162.
- [7] 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 Tech.*, vol. 49, no. 3, pp. 480–490, Mar. 2001.
- [8] 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 Tech.*, vol. 54, no. 7, pp. 3060–3068, Jul. 2006.
- [9] W. Yu and Z. Wang, "Enhanced QMM-BEM solver for three-dimensional multiple-dielectric capacitance extraction within the finite domain," *IEEE Trans. Microw. Theory Tech.*, vol. 52, no. 2, pp. 560–566, Feb. 2004.
- [10] S. Börm, L. Grasedyck, and W. Hackbusch, *Hierarchical Matrices*, ser. Lecture Note 21. Bonn, Germany: Max Planck Inst. Math., 2003.
- [11] W. Hackbusch and B. Khoromskij, "A sparse matrix arithmetic based on \mathcal{H} -matrices. Part I: Introduction to \mathcal{H} -matrices," *Computing*, vol. 62, pp. 89–108, 1999.
- [12] W. Hackbusch and B. N. Khoromskij, "A sparse \mathcal{H} -matrix arithmetic. Part II: Application to multi-dimensional problems," *Computing*, vol. 64, pp. 21–47, 2000.
- [13] S. Börm and W. Hackbusch, " \mathcal{H}^2 -matrix approximation of integral operators by interpolation," *Appl. Numer. Math.*, vol. 43, pp. 129–143, 2002.
- [14] S. Börm, "Approximation of integral operators by \mathcal{H}^2 -matrices with adaptive bases," *Computing*, vol. 74, pp. 249–271, 2005.
- [15] S. Börm, " \mathcal{H}^2 -matrices—Multilevel methods for the approximation of integral operators," *Comput. Visual. Sci.*, vol. 7, pp. 173–181, 2004.
- [16] S. Börm, " \mathcal{H}^2 -matrix arithmetics in linear complexity," *Computing*, vol. 77, pp. 1–28, 2006.
- [17] W. Chai and D. Jiao, "An \mathcal{H}^2 -matrix-based integral-equation solver of linear-complexity for large-scale full-wave modeling of 3-D circuits," in *IEEE 17th Elect. Perform. Electron. Packag. Conf.*, Oct. 2008, pp. 283–286.
- [18] W. Chai and D. Jiao, "An \mathcal{H}^2 -matrix-based integral-equation solver of reduced complexity and controlled accuracy for solving electrodynamic problems," *IEEE Trans. Antennas Propag.*, vol. 57, no. 10, pp. 3147–3159, Oct. 2009.
- [19] W. Chai, D. Jiao, and C. C. Koh, "A direct integral-equation solver of linear complexity for large-scale 3-D capacitance and impedance extraction," in *46th ACM/EDAC/IEEE DAC*, Jul. 2009, pp. 752–757.
- [20] HiCap. Texas A&M Univ., College Station, TX, Feb. 2010. [Online]. Available: <http://dropzone.tamu.edu~wshi/pub.html>
- [21] H. Boltz, "Matrix inversion lemma," *Wikipedia* 2011. [Online]. Available: http://en.wikipedia.org/wiki/Invertible_matrix

- [22] J. Shaeffer, "Direct solve of electrically large integral equations for problem sizes to 1 M unknowns," *IEEE Trans. Antennas Propag.*, vol. 56, no. 8, pp. 2306–2313, Aug. 2008.
- [23] M. Bebendorf and W. Hackbusch, "Existence of \mathcal{H} -matrix approximations to the inverse fe-matrix of elliptic operators with L^∞ -coefficients," *Numer. Math.*, vol. 95, pp. 1–28, 2003.
- [24] R. J. Adams, Y. Xu, X. Xu, S. D. Gedney, and F. X. Canning, "Modular fast direct electromagnetic analysis using local-global solution modes," *IEEE Trans. Antennas Propag.*, vol. 56, no. 8, pp. 2427–2441, Aug. 2008.
- [25] L. Greengard, D. Gueyffier, P.-G. Martinsson, and V. Rokhlin, "Fast direct solvers for integral equations in complex three-dimensional domains," *Acta Numer.*, pp. 261–288, 2009.
- [26] W. Chai and D. Jiao, "A complexity-reduced h-matrix based direct integral equation solver with prescribed accuracy for large-scale electrodynamic analysis," in *IEEE Int. Antennas Propag. Symp.*, Jul. 2010., (see also Purdue ECE Tech. Rep. [Online]. Available: <http://docs.lib.purdue.edu/ecetr/411>).



Wenwen Chai (S'09–M'11) received the B.S. degree from the University of Science and Technology of China, Hefei, China, in 2004, the M.S. degree from the Chinese Academy of Sciences, Beijing, China, in 2007, both in electrical engineering, and is currently working toward the Ph.D. degree at Purdue University, West Lafayette, IN.

She is currently with the School of Electrical and Computer Engineering, Purdue University, as a member of the On-Chip Electromagnetics Group. Her research is focused on computational

electromagnetics, high-performance very large scale integration (VLSI) computer-aided design (CAD), and fast and high-capacity numerical methods.

Ms. Chai was the recipient of the IEEE Antennas and Propagation Society Doctoral Research Award for 2009–2010.



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

She was then a Senior Computer-Aided Design (CAD) Engineer, Staff Engineer, and Senior Staff Engineer with the Technology CAD Division, Intel Corporation, until September 2005. In September 2005, she joined Purdue University, West Lafayette, IN, as an Assistant Professor with the School of Electrical and Computer Engineering. In 2009, she became a Tenured Associate Professor. She has

authored two book chapters and over 140 papers in refereed journals and international conferences. Her current research interests include computational electromagnetics, high-frequency digital, analog, mixed-signal, and 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 bio-electromagnetics.

Dr. Jiao was among 100 engineers chosen for the National Academy of Engineering's 2011 U.S. Frontiers of Engineering Symposium. She was the recipient of the 2010 Ruth and Joel Spira Outstanding Teaching Award, the 2008 National Science Foundation (NSF) CAREER Award, the 2006 Jack and Cathie Kozik Faculty Start Up Award (which recognizes an outstanding new faculty member with the Electrical and Computer Engineering (ECE) Staff, Purdue University), the Office of Naval Research (ONR) Award under Young Investigator Program in 2006, the 2004 Best Paper Award of the Intel Corporation Annual Corporate-Wide Technology Conference (Design and Test Technology Conference) for her work on generic broadband model of high-speed circuits, the 2003 Intel Corporation Logic Technology Development (LTD) Divisional Achievement Award in recognition of her work on the industry-leading Broad-Spice modeling/simulation capability for designing high-speed microprocessors, packages, and circuit boards, the Intel Corporation Technology CAD Divisional Achievement Award for the development of innovative full-wave solvers for high-frequency IC design, the 2002 Intel Corporation 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 LTD Team Quality Award for her outstanding contribution to the development of the measurement capability and simulation tools for high-frequency on-chip crosstalk, and the 2000 Raj Mittra Outstanding Research Award of the University of Illinois at Urbana-Champaign.