Efficient Linear System Solution Techniques in the Simulation of Large Dense Mutually Inductive Circuits

DOI:
10.1109/ICCD46524.2019.00063

Document Version
Accepted author manuscript

Published in:
Proceedings - 2019 IEEE International Conference on Computer Design, ICCD 2019

Efficient Linear System Solution Techniques in the Simulation of Large Dense Mutually Inductive Circuits

Charalampos Antoniadis*, Milan Mihajlovic†, Nestor Evmorfopoulos*, Georgios Stamoulis* and Vasilis F. Pavlidis†
*Dept. of Electrical & Computer Engineering, University of Thessaly, Volos, Greece
{haadonia, nestevmo, georges}@e-ce.uth.gr
†School of Computer Science, The University of Manchester, Manchester, UK
{milan, pavlidis}@cs.man.ac.uk

***Abstract*—The verification of integrated Circuits (ICs) in deep submicron technologies requires that all mutual inductive effects are taken into account to properly validate the performance and reliable operation of the chip. However, the inclusion of all mutual inductive couplings results in a fully dense inductance matrix that renders the circuit simulation computationally prohibitive. In this paper, we present efficient techniques for the solution of the linear systems arising in transient analysis of large mutually inductive circuits. These techniques involve the compression of the dense inductance matrix block by low-rank products in hierarchical matrix format, as well as the development of a Schur-complement preconditioner for the iterative solution of the transient linear system (which comprises sparse blocks alongside the dense inductance block). Experimental results indicate that substantial compression rates of the inductance matrix can be achieved without compromising accuracy, along with considerable reduction in iteration counts and execution time of iterative solution methods.**

***Index Terms*—RLC Simulation, mutual inductance, Hierarchical Matrices, Preconditioning, Schur Complement, Krylov methods**

I. Introduction

In high-frequency integrated circuits, parasitic inductance becomes prominent and needs to be considered in simulation to accurately predict the behavior and verify the reliability of the chip. However, the simulation of large RLC parasitic networks can be extremely time-consuming or, in some cases, even infeasible due to their sheer size.

In the past, it has been sufficient to include only self-inductance in the RLC simulation, resulting in large linear systems with sparse coefficient matrices. For such systems, iterative solvers are the methods of choice offering small memory requirements and excellent scalability. However, in modern high-frequency ICs there is an increasing demand for modeling all mutual inductive couplings between the different circuit blocks, leading to a fully dense inductance matrix that affects the scaling properties and storage requirements of iterative solvers. As the convergence rate of iterative methods cannot be predicted *a-priori*, the use of appropriate preconditioners is necessary to ensure a robust and fast convergence across a wide range of problems.

As direct sparsification (by truncation) of the dense inductance matrix is well-known to lead to loss of circuit passivity and simulation stability [1], most previous attempts in the literature have focused on sparsifying its inverse (called the reluctance matrix), which is amenable to sparsification due to its diagonal dominance [2] [3] [4]. However, this requires prior inversion of a large dense matrix which is a very expensive computational procedure, making the reluctance-based approaches feasible only for problems of moderate sizes. An approach that avoids the expensive inversion of the inductance matrix, by computing selectively only some of the entries of the reluctance matrix up to a given sparsity ratio, has been proposed in [5]. However, this approach introduces error that is difficult to quantify in advance, thus limiting its application in practical settings.

In this paper, we propose the compression (instead of sparsification) of the actual inductance matrix via the approximation of large off-diagonal blocks by low-rank products, in a scheme known as hierarchical matrix (or $\mathcal{H}$-matrix) format. This format conserves memory by storing only the low-rank factors of the blocks, while enabling the execution of basic matrix-vector operations required by iterative solvers in near optimal complexity. Moreover, we propose a block preconditioner based on an efficient approximation of the Schur complement, which improves considerably the convergence rate of iterative methods and has inexpensive application. Experimental results demonstrate significant reduction in memory footprint and execution time with negligible loss of accuracy, in comparison to existing dense linear solvers in typical simulation scenarios.

The rest of the paper is organized as follows: Section II offers an overview of transient RLC analysis, iterative solvers and hierarchical matrix format. Section III introduces hierarchical matrices as an approximation for the dense inductance matrix. Section IV presents the proposed preconditioning scheme for the simulation of mutually inductive RLC circuits. Section V provides experimental results that demonstrate the efficiency of the proposed methodology, while conclusions are drawn in section VI.

II. Background

A. Overview of RLC transient analysis

Consider an RLC circuit composed of n nodes and m inductive branches with mutual inductive coupling between them, as well as its Modified Nodal Analysis (MNA) description [6]:

$$\widetilde{\mathbf{G}}\mathbf{x}(t) + \widetilde{\mathbf{C}}\dot{\mathbf{x}}(t) = \widetilde{\mathbf{e}}(t) \tag{1}$$

where

$$\widetilde{\mathbf{G}} = \begin{bmatrix} \mathbf{G} & \mathbf{A}_L \\ -\mathbf{A}_L^T & \mathbf{0} \end{bmatrix}, \widetilde{\mathbf{C}} = \begin{bmatrix} \mathbf{C} & \mathbf{0} \\ \mathbf{0} & \mathbf{L} \end{bmatrix}, \mathbf{x}(t) = \begin{bmatrix} \mathbf{v}(t) \\ \mathbf{i}(t) \end{bmatrix}, \widetilde{\mathbf{e}}(t) = \begin{bmatrix} \mathbf{e}(t) \\ \mathbf{0} \end{bmatrix}$$

In the above, $\mathbf{G} \in \Re^{n\times n}$ and $\mathbf{C} \in \Re^{n\times n}$ are the node conductance and node capacitance matrices respectively, $\mathbf{L} \in \Re^{m\times m}$ is the dense inductance matrix (with self-inductances as diagonal entries and mutual inductances as off-diagonal entries), $\mathbf{A}_L \in \Re^{n\times m}$ is the corresponding node-to-branch incidence matrix, $\mathbf{v}(t) \in \Re^n$ and $\mathbf{i}(t) \in \Re^m$ are the vectors of the unknown node voltages and branch currents, and $\mathbf{e}(t) \in \Re^n$ is the vector of excitations from independent sources at the nodes (assuming, without loss of generality, that voltage sources have been transformed to Norton-equivalent current sources). Applying the Backward-Euler numerical integration method in (1), we arrive at the problem of solving a system of $n+m$ linear algebraic equations at each discrete time $t_k, k = 1, 2, \ldots$ (starting with initial values $\mathbf{x}(t_0) = [\mathbf{v}(0) \quad \mathbf{i}(0)]^T$):

$$\mathbf{J}_k\mathbf{x}(t_k) = \mathbf{b}(t_k) \tag{2}$$

where

$$\mathbf{J}_k = \begin{bmatrix} \frac{1}{h_k}\mathbf{C} + \mathbf{G} & \mathbf{A}_L \\ -\mathbf{A}_L^T & \frac{1}{h_k}\mathbf{L} \end{bmatrix}, \quad \mathbf{b}(t_k) = \widetilde{\mathbf{e}}(t_k) + \frac{\widetilde{\mathbf{C}}}{h_k}\mathbf{x}(t_{k-1})$$

and $h_k = t_k - t_{k-1}$, $k = 1, 2, \ldots$ is the chosen time-step size (which can be either fixed or variable during the analysis). The above is a linear system of the form $\mathbf{Ax} = \mathbf{b}$ that has to be solved at every discrete time t_k, $k = 1, 2, \ldots$.

B. Iterative linear solvers

Direct methods (based on matrix factorization) are not practically feasible for solving large dense systems or systems with a large dense block (like $\frac{1}{h_k}\mathbf{L}$ in (2)), due to their excessive runtime and memory requirements. The only viable option for such systems is the use of iterative methods, and particularly Krylov-subspace iterative methods like GMRES (which is suitable for general unsymmetric systems like (2)) [7]. The operation with the dominant cost inside the iteration loop of Krylov-subspace methods (and GMRES in particular) is the matrix-vector multiplication, which for the system (2) can be dealt efficiently by the hierarchical matrix framework that is introduced in the next subsection and applied to the inductance matrix $\mathbf{L}$ in Section III. Other operations of Krylov-subspace iterative methods like inner products, scalar-vector products and vector additions are not expensive computationally.

The convergence rate of Krylov-subspace methods is determined by the spread of the eigenvalues of the system matrix and their distance from 1 [8]. In particular, convergence is fast when the eigenvalues are tightly clustered together and slow when they are spread apart. A slow convergence rate can be alleviated by using a preconditioner matrix $\mathbf{M}$ and to solve an equivalent system $\mathbf{M}^{-1}\mathbf{Ax} = \mathbf{M}^{-1}\mathbf{b}$.

An effective preconditioner needs to satisfy the following two prerequisites:

- Preconditioning should lead to a much tighter clustering of the eigenvalues of $\mathbf{M}^{-1}\mathbf{A}$ than those of the original matrix $\mathbf{A}$, thus reducing significantly the iteration counts.
- The reduction in the number of iterations should offset the computational overhead introduced by the solution of $\mathbf{Mz}_j = \mathbf{r}_j$ in every iteration.

C. Low-rank product approximation and hierarchical matrices

If $\mathbf{A} \in \Re^{n\times n}$ is a square matrix or matrix block with Singular Value Decomposition (SVD) $\mathbf{A} = \mathbf{U\Sigma V}^T$, where $\mathbf{\Sigma} = diag(\sigma_1, \ldots, \sigma_n)$ and $\sigma_1 > \cdots > \sigma_n$, then by introducing the following partition:

$$\mathbf{U} = [\mathbf{U}_1 \quad \mathbf{U}_2], \quad \mathbf{\Sigma} = \begin{bmatrix} \mathbf{\Sigma}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{\Sigma}_2 \end{bmatrix}, \quad \mathbf{V}^T = \begin{bmatrix} \mathbf{V}_1^T \\ \mathbf{V}_2^T \end{bmatrix}$$

with $\mathbf{U}_1 \in \Re^{n\times r}$, $\mathbf{\Sigma}_1 \in \Re^{r\times r}$ and $\mathbf{V}_1^T \in \Re^{r\times n}$, the optimal low-rank product approximation of rank-r of $\mathbf{A}$ is defined as $\widetilde{\mathbf{A}} = (\mathbf{U}_1\mathbf{\Sigma}_1^{1/2})(\mathbf{V}_1\mathbf{\Sigma}_1^{1/2})^T \equiv \mathbf{ZY}^T$. It has been proven in [9] that this approximation satisfies the optimization problem $\min_{\widetilde{\mathbf{A}}} \|\mathbf{A} - \widetilde{\mathbf{A}}\|$ s.t. $\text{rank}(\widetilde{\mathbf{A}}) = r$ for any common matrix norm. The benefit of the factorization $\widetilde{\mathbf{A}} = \mathbf{ZY}^T$ with $\mathbf{Z} = \mathbf{U}_1\mathbf{\Sigma}_1^{1/2}$ and $\mathbf{Y} = \mathbf{V}_1\mathbf{\Sigma}_1^{1/2}$ is that only the factors $\mathbf{Z}$ and $\mathbf{Y}$ have to be kept in memory instead of the whole $n\times n$ matrix $\mathbf{A}$ (see Fig. 1a). The above low-rank product approximation can be straightforwardly extended to rectangular matrix blocks.

Hierarchical matrices or $\mathcal{H}$-matrices [10] are a lossy compressed matrix format which relies on the partitioning of a dense matrix into a number of sub-matrix blocks that can be approximated efficiently and accurately by low-rank products. The special structure of $\mathcal{H}$-matrices allows the development of algorithms for the basic operations of matrix-vector multiplication and matrix factorization with near optimal asymptotic complexity.

III. Approximation of the Inductance Matrix with Hierarchical Matrices

Because of the natural fact that interconnect segments which are farther apart exhibit weaker mutual inductive interactions, the inductance matrix $\mathbf{L}$ will be characterized by progressively smaller off-diagonal elements while moving away from the diagonal (assuming that segments in close physical proximity are enumerated consecutively - otherwise suitable permutation matrices can be applied). Then the matrix blocks that are away from the diagonal can be efficiently approximated by low-rank products and the whole inductance matrix by an appropriate $\mathcal{H}$-matrix (see Fig. 1b). The size and the number of blocks, as well as the order of the low-rank approximation of each block, constitute trade-off parameters between the degree of compression and the quality of approximation.

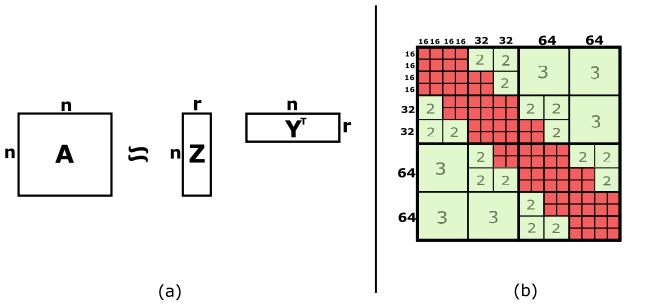


Fig. 1: (a) Approximation of a $n \times n$ dense block with a low-rank product. Only $\mathcal{O}(2rn)$ storage is required, while parameter r controls the accuracy of the approximation. (b) Example of 256×256 inductance matrix in $\mathcal{H}$-matrix format. Blocks around diagonal (red-colored blocks) correspond to mutual inductances in close physical proximity to each other. Blocks away form the diagonal (green-colored blocks) have progressively smaller numerical values and can be approximated by low-rank products (the rank of each block is indicated inside the block).

It is noted that the usage of $\mathcal{H}$-matrix format does not require the *a-priori* knowledge of the whole inductance matrix. Instead, only the spatial arrangement of the interconnects, which is available before the assembly of the inductance matrix, is required to perform the matrix blocking. Thus, provided that we integrate the $\mathcal{H}$-matrix library with the inductance extraction tool, we can approximate a matrix block by a low-rank product immediately after its computation, and next store it directly in $\mathcal{H}$-matrix format rather than as part of a dense inductance matrix.

IV. Solution and Preconditioning of the Transient Linear System

A. Multiplication of transient system matrix with a vector

The matrix $\mathbf{J}_k$ is composed of different storage formats, with $\mathbf{G} + \frac{1}{h_k}\mathbf{C}$ and $\mathbf{A}_L$ being in sparse format (compressed row or column form) and $\mathbf{L}$ in $\mathcal{H}$-matrix format. Thus, the multiplication of $\mathbf{J}_k$ with a vector inside the iteration loop of a Krylov-subspace method like GMRES, can be performed in a block fashion by calling the appropriate sparse and $\mathcal{H}$-matrix subroutines.

B. Preconditioner formulation

Consider the block LU factorization of $\mathbf{J}_k$ of (2):

$$\mathbf{J}_k = \mathbf{L_J U_J} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ -\mathbf{A}_L^T(\mathbf{G} + \frac{1}{h_k}\mathbf{C})^{-1} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{G} + \frac{1}{h_k}\mathbf{C} & \mathbf{A}_L \\ \mathbf{0} & \mathbf{S} \end{bmatrix}$$

where

$$\mathbf{S} = \frac{1}{h_k}\mathbf{L} + \mathbf{A}_L^T(\mathbf{G} + \frac{1}{h_k}\mathbf{C})^{-1}\mathbf{A}_L$$

is the Schur complement of $\mathbf{J}_k$. Because $\mathbf{L_J}$ is a block lower triangular matrix with identity blocks on the main diagonal, and thus has all eigenvalues equal to 1, the block upper triangular matrix

$$\mathbf{U_J} = \begin{bmatrix} \mathbf{G} + \frac{1}{h_k}\mathbf{C} & \mathbf{A}_L \\ \mathbf{0} & \mathbf{S} \end{bmatrix} \tag{3}$$

constitutes an ideal preconditioner for $\mathbf{J}_k$ (meaning that GMRES will converge in only one iteration).

C. Preconditioner application

The preconditioning step in the body of Krylov methods involves the solution of the following linear system:

$$\mathbf{U_J z} = \mathbf{r} \Rightarrow \begin{bmatrix} \mathbf{G} + \frac{1}{h_k}\mathbf{C} & \mathbf{A}_L \\ \mathbf{0} & \mathbf{S} \end{bmatrix} \begin{bmatrix} \mathbf{z}_1 \\ \mathbf{z}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{r}_1 \\ \mathbf{r}_2 \end{bmatrix} \quad (4)$$

The system (4) has a block upper triangular coefficient matrix and its solution can be achieved by block back substitution. Specifically, we first solve $\mathbf{S}\mathbf{z}_2 = \mathbf{r}_2$ (details are given in the next paragraph), then update the right hand side as $\mathbf{r}_1 = \mathbf{r}_1 - \mathbf{A}_L\mathbf{z}_2$, and finally solve $(\mathbf{G}+\frac{1}{h_k}\mathbf{C})\mathbf{z}_1 = \mathbf{r}_1$. The latter is a $n \times n$ sparse linear system which can be solved by a direct or iterative sparse linear solver. Since it can be demonstrated that $\mathbf{G}+\frac{1}{h_k}\mathbf{C}$ is a symmetric diagonally dominant matrix with non-positive off-diagonal elements [11], it is recommended to use iterative methods for which very efficient preconditioners have been developed [12], [13].

The Schur complement $m \times m$ system $\mathbf{S}\mathbf{z}_2 = \mathbf{r}_2$ has coefficient matrix with two additive terms, $\mathbf{S}_1 \equiv \frac{1}{h_k}\mathbf{L}$ and $\mathbf{S}_2 \equiv \mathbf{A}_L^T(\mathbf{G} + \frac{1}{h_k}\mathbf{C})^{-1}\mathbf{A}_L$, both of which are dense and in different matrix formats ($\mathbf{S}_1$ is stored as an $\mathcal{H}$-matrix). However, since the relative contributions of $\mathbf{S}_1$ and $\mathbf{S}_2$ depend on the time step size h_k, we can choose one term over the other for the typical range of step sizes in a variable-step simulation. Specifically, for small step sizes the term $\mathbf{S}_1 = \frac{1}{h_k}\mathbf{L}$ dominates and the system $\frac{1}{h_k}\mathbf{L}\mathbf{z}_2 = \mathbf{r}_2$ can be solved by LU factorization of $\mathbf{L}$ in $\mathcal{H}$-matrix format. For larger step sizes, the term $\mathbf{S}_2 = \mathbf{A}_L^T(\mathbf{G} + \frac{1}{h_k}\mathbf{C})^{-1}\mathbf{A}_L$ is dominant and the block preconditioner becomes

$$\begin{bmatrix} \mathbf{G} + \frac{1}{h_k}\mathbf{C} & \mathbf{A}_L \\ \mathbf{0} & \mathbf{A}_L^T(\mathbf{G} + \frac{1}{h_k}\mathbf{C})^{-1}\mathbf{A}_L \end{bmatrix} \begin{bmatrix} \mathbf{z}_1 \\ \mathbf{z}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{r}_1 \\ \mathbf{r}_2 \end{bmatrix}$$

which can be straightforwardly derived to be equivalent to the system

$$\begin{bmatrix} \mathbf{G} + \frac{1}{h_k}\mathbf{C} & \mathbf{A}_L \\ -\mathbf{A}_L^T & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{z}_1 \\ \mathbf{z}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{r}_1 \\ \mathbf{r}_2 \end{bmatrix} \quad (5)$$

The latter is an $(n+m)\times(n+m)$ sparse linear system which can be solved by any direct or iterative linear solver.

D. Selection of the dominant term of Schur complement

The choice of the approximation of $\mathbf{S}$ as $\mathbf{S}_1$ or $\mathbf{S}_2$, for a given timestep, can be guided by quantifying the relative magnitude of each term in its spectral norm (see Algorithm 1). While there exist efficient routines to compute the spectral norm of $\frac{1}{h_k}\mathbf{L}$ in $\mathcal{H}$-matrix format, the spectral norm of $\mathbf{A}_L^T(\mathbf{G} + \frac{1}{h_k}\mathbf{C})^{-1}\mathbf{A}_L$ can be estimated by the inequality

Algorithm 1 Choice of suitable approximation of $\mathbf{S}$ for a given timestep h_k

function $\mathbf{S}$ = APPROX($\frac{1}{h_k}\mathbf{L}$, $\mathbf{A}_L^T(\mathbf{G} + \frac{1}{h_k}\mathbf{C})^{-1}\mathbf{A}_L$)
  if $\left\| \frac{1}{h_k}\mathbf{L} \right\|_{sp} > \left\| \mathbf{A}_L^T(\mathbf{G} + \frac{1}{h_k}\mathbf{C})^{-1}\mathbf{A}_L \right\|_{sp}$ **then**
    $\mathbf{S} \simeq \frac{1}{h_k}\mathbf{L}$
  else
    $\mathbf{S} \simeq \mathbf{A}_L^T(\mathbf{G} + \frac{1}{h_k}\mathbf{C})^{-1}\mathbf{A}_L$
  end if
end function

$$||\mathbf{A}_L^T(\mathbf{G} + \frac{1}{h_k}\mathbf{C})^{-1}\mathbf{A}_L||_{sp} \leq ||\mathbf{A}_L||_2||(\mathbf{G} + \frac{1}{h_k}\mathbf{C})^{-1}||_{sp}||\mathbf{A}_L^T||_2 \quad (6)$$

Due to the specific structure of the interpolation matrix $\mathbf{A}_L$ the inequality (6) is fairly sharp and the estimate is getting better as h_k increases. With $||\mathbf{A}_L||_2 = \sqrt{2}$ we have

TABLE I: Details of the experimental benchmarks

Benchmark	Number of RL branches (m)	Number of circuit nodes (n)
bus1	256	544
bus2	1024	2080
bus3	4096	8208
bus4	8192	16416
bus5	16384	32832

TABLE II: Computational impact of the approximation of dense inductance matrix $\mathbf{L}$ with $\mathcal{H}$-matrix $\mathbf{L}_\mathcal{H}$.

Bench.	Compr. Time to $\mathcal{H}$ matrix	Storage			Mat-vec Product Time		
		Dense L	$\mathcal{H}$ matrix	Mem. save	Dense L	$\mathcal{H}$ matrix	Speed up
bus1	20ms	512.17kB	144.26kB	71.8%	0.04ms	0.03ms	1.33×
bus2	30ms	8MB	746.65kB	90.8%	0.65ms	0.6ms	10.8×
bus3	180ms	128MB	4.89MB	96.1%	11ms	0.32ms	34.4×
bus4	470ms	512MB	15.96MB	96.8%	50ms	1.7ms	29.5×
bus5	520ms	2GB	50.64MB	97.5%	225ms	8ms	28.1×

$$||\mathbf{A}_L^T(\mathbf{G} + \frac{1}{h_k}\mathbf{C})^{-1}\mathbf{A}_L||_{sp} \leq 2||(\mathbf{G} + \frac{1}{h_k}\mathbf{C})_h^{-1}||_{sp} = \frac{2}{\lambda_{min}(\mathbf{G} + \frac{1}{h_k}\mathbf{C})} \quad (7)$$

where $\lambda_{min}(\mathbf{G} + \frac{1}{h_k}\mathbf{C})$ is the minimum eigenvalue of the matrix $\mathbf{G}+\frac{1}{h_k}\mathbf{C}$. Its estimate can be obtained efficiently by the inverse power iteration [14] where the matrix-vector product from the normal power method is replaced by the solution of a linear system.

V. Experimental Results

A. Experimental setup

For the experimental evaluation of the compression rates obtained by storing the inductance matrix $\mathbf{L}$ as a $\mathcal{H}$-matrix and the efficiency of the proposed preconditioning method we created a set of interconnect RLC models with inductive and capacitive coupling. The inductance matrix, with all mutual inductances, was assembled with FastHenry [15], while we assume that there is a capacitive path from all nodes to the ground and a capacitive coupling between adjacent nodes. Structural details of the benchmarks are summarized in Table I. The simulation of the RLC interconnect models was performed with a driver resistance of 30Ω, load capacitance of 20fF, total wire self-capacitance of 40fF and total coupling capacitance between adjacent wires of 20fF. A 1V 20ps ramp voltage source was applied to one of the wires while the remaining were kept inactive. Our experimental framework was developed in C/C++ and all our experiments were performed on a system with a 3.6GHz Intel Core i7 CPU and 16GB memory. It is noted that since we used an off-the-shelf RLC extractor like FastHenry (rather than develop a custom extraction tool), we were forced to store the whole dense inductance matrix in memory before converting it to $\mathcal{H}$-matrix format, and thus were restricted in the largest circuit that could be handled by the memory of the target machine.

B. Efficiency of the compression of inductance matrix by $\mathcal{H}$-matrices

The $\mathcal{H}$-matrix approximation requires information on the geometrical coordinates of the interconnect branches, but it is pointed out that the physical structure of the circuit model can be arbitrary and is not restricted to specific configurations like parallel buses of conductors. In order to compress the dense inductance matrix by $\mathcal{H}$-matrices we adopt in this work the HLIBpro library [16] [17] [18]. HLIBpro offers the approximation routines to store a dense matrix in the $\mathcal{H}$-matrix format and perform a set of matrix operations with $\mathcal{H}$-matrices. The inputs to HLIBpro are the geometrical coordinates of the nodes of the physical structure. HLIBpro identifies the node indices that correspond to closer interconnects and performs a segmentation of $\mathbf{L}$ into blocks. For the clustering process of the coordinates into groups and, in turn, the segmentation of $\mathbf{L}$ into a number of blocks of certain size we used the default set of

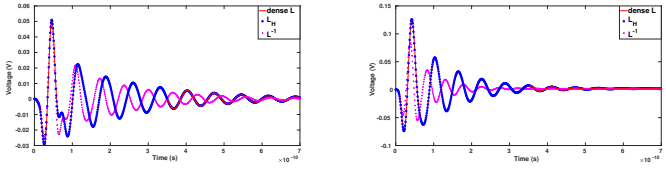
TABLE III: Iteration count to solve the transient system (2) with the proposed preconditioner and without preconditioner for timesteps 10fs, 1ps, 100ps at one time instant. The spectral norms of the Schur complement terms are also reported for each timestep.

Bench.	h_k = 10fs				h_k = 1ps				h_k = 100ps			
	# iter. with prec. $\mathbf{U_J}$	# iter. w/o prec.	$\|\|\mathbf{S}_1\|\|_{sp}$	$\|\|\mathbf{S}_2\|\|_{sp}$	# iter. with prec. $\mathbf{U_J}$	# iter w/o prec.	$\|\|\mathbf{S}_1\|\|_{sp}$	$\|\|\mathbf{S}_2\|\|_{sp}$	# iter. with prec. $\mathbf{U_J}$	# iter. w/o prec.	$\|\|\mathbf{S}_1\|\|_{sp}$	$\|\|\mathbf{S}_2\|\|_{sp}$
bus1	2	2021	**3.7e+5**	20	21	2043	**3.7e+3**	1.6e+3	3	NA	37.4	**1.6e+5**
bus2	2	1848	**1e+5**	64.6	16	3961	1e+3	**6.3e+3**	3	NA	10.4	**6.3e+5**
bus3	7	1326	**7.1e+3**	494.8	13	4476	71.5	**4.9e+4**	4	NA	0.7	**4.9e+6**
bus4	7	2427	**1.3e+4**	505.8	18	5563	129.9	**5e+4**	4	NA	1.3	**5e+6**
bus5	7	1876	**2.5e+4**	509	17	6158	250	**5.1e+4**	4	NA	2.5	**5.1e+6**

parameters of HLIBpro. Each block $\mathbf{A}_i$ was approximated by low-rank product $\widetilde{\mathbf{A}}_i = \mathbf{Z}_i \mathbf{Y}_i^T$ such that $\|\mathbf{A}_i - \widetilde{\mathbf{A}}_i\|_F \leqslant 10^{-4}$. Table II reports the time required to perform $\mathcal{H}$-matrix compression of $\mathbf{L}$, as well as the memory savings and the speedups of matrix-vector multiplication required in the Krylov loop.

C. Preconditioner efficiency analysis

The efficiency of the proposed preconditioner depends on the timestep h_k used in the simulation. Table III reports the iteration count to solve (2) with the proposed preconditioner, which uses two different approximations of the Schur complement ($\mathbf{S} = \mathbf{S}_1$ and $\mathbf{S} = \mathbf{S}_2$), and with no preconditioner for the range of time step sizes that is of practical interest. Also, it reports the spectral norm of $\mathbf{S}_1$ and $\mathbf{S}_2$ used to approximate $\mathbf{S}$ in (4) for a given timestep, as in Algorithm 1. We adopt $h_k \in \{10\text{fs}, 1\text{ps}, 100\text{ps}\}$, where the lower range is used to compute detailed waveforms and the upper range is used to either examine the general trend in waveforms or to compute steady state. From the iteration count we can observe that the for small steps sizes $h_k \sim 10$ fs the choice $\mathbf{S} = \mathbf{S}_1$ is superior, and for $h_k \sim$ [1 100]ps the choice $\mathbf{S} = \mathbf{S}_2$ leads to smaller iteration count. This is in agreement with the discussion in Subsection IV-C on different weightings of the two terms $\mathbf{S}_1$ and $\mathbf{S}_2$ to the Schur complement of $\mathbf{S}$. The cross-over point between the two preconditioners occurs approximately in the interval [10fs 1ps]. In the case with no preconditioning, the Krylov-iterative method failed to converge for larger step sizes.



(a) (b)

Fig. 2: Voltage response at randomly chosen nodes of bus1 (a) and bus3 (b) with h_k =1ps (the choice of h_k = 1ps shows all the details in the response waveform) obtained by full SPICE, $\mathcal{H}$-matrix approximation with preconditioning, and sparse reluctance simulation with same memory as $\mathcal{H}$-matrix. The proposed approach is indistinguishable from SPICE, while reluctance-based simulation exhibits significant deviation.

D. Transient analysis results

Combining the benefits of the storage of the dense inductance matrix as an $\mathcal{H}$-matrix and the reduced number of Krylov iterations through the proposed preconditioner, Table IV reports the speedups of the simulation of RLC benchmarks for h_k=10ps and 30 time points, in comparison to standard SPICE simulation (dense $\mathbf{L}$) and simulation with H-matrix L but without preconditioner. For this simulation scenario, the proposed methodology leads to speedups up to 2139× for the largest circuit. Regarding the accuracy of the $\mathcal{H}$-matrix approximation, the relative rms error of the simulation compared to exact SPICE was less than 0.01 in all nodes of every benchmark. Simulation waveforms for random nodes of two benchmark circuits are graphically displayed in Fig. 2, where it can be observed that the waveforms for $\mathcal{H}$-matrix with the proposed preconditioner are indistinguishable from exact SPICE. Superimposed in the same figures are waveforms from sparse reluctance-based simulation, obtained by inversion and truncation of $\mathbf{L}$ (with sparsity ratio leading to same memory footprint as the H-matrix approximation),

TABLE IV: Runtime results of the whole simulation of benchmark RLC circuits with timestep 1ps.

Bench.	Size of Matrix $\mathbf{J}_k$	SPICE Time	Simulation with $\mathcal{H}$-matrix w/o preconditioner		Simulation with $\mathcal{H}$-matrix with preconditioner	
			Time	Speedup	Time	Speedup
bus1	800	1.1s	1s	3.75×	0.28s	3.9×
bus2	3104	61s	9.75s	6.21×	1.57s	38.8×
bus3	12304	4989s	110s	19.19×	5.72s	872×
bus4	24608	31917s	515s	22.4×	23s	1388.3×
bus5	49216	194694s	2545s	76.5×	91s	2139×

which can be observed to exhibit a clear deviation from both exact SPICE and the proposed methodology. Note that for the largest benchmark circuit, the inversion of the dense $\mathbf{L}$ (with size 16.3K) took an additional 111s.

VI. Conclusion

In this paper, we use the $\mathcal{H}$-matrices for the compression of the dense inductance matrix, arising in the modelling of all mutual inductive couplings between the interconnects in ICs and we present an efficient preconditioner for the system to be solved during the time integration. With these two ingredients, we can tackle much larger problems than previously possible on a given computing platform, both in terms of storage and wall clock time. Our experimental results indicate that a very good compression ratio of inductance matrix can be attained without compromising accuracy, while the proposed preconditioner reduces the iterations count of Krylov iterative method in simulation, effectively leading to a significant speedup of the simulation.

References

[1] Z. He et al., *SPIE: Sparse Partial Inductance Extraction*, Design Automation Conf. (DAC), 1997.
[2] C. Antoniadis et al., *On the Sparsification of the Reluctance Matrix in RLCk Circuit Transient Analysis*, Int. Conf. Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD), 2018.
[3] S. Zeng et al., *Efficient partial reluctance extraction for large-scale regular power grid structures*, IEICE Transactions Fundamentals of Electronics Communications and Computer Sciences, vol. 92-A, pp. 1476-1484, 2009.
[4] S. Zeng et al., *Efficient Power Network Analysis with Modeling of Inductive Effects*, IEICE Transactions Fundamentals of Electronics Communications and Computer Sciences, vol. 93-A, pp. 1196-1203, 2010.
[5] I. Apostolopoulou et al., *Selective inversion of inductance matrix for large-scale sparse RLC simulation*, Design Automation Conf. (DAC), 2014.
[6] F. Najm, *Circuit Simulation*, Wiley-IEEE Press, 2010.
[7] R. Barrett et al., *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*, SIAM, 1994.
[8] Saad Y., *Iterative Methods for Sparse Linear Systems*, SIAM, 2003.
[9] C. Eckart. et al., *The approximation of one matrix by another of lower rank*, Psychometrika, vol. 1, pp. 211-218, 1936.
[10] W. Hackbush, *Hierarchical Matrices*, Springer, 2015.
[11] A. Ruehli, *Circuit Analysis, Simulation & Design*, North-Holland, 1986.
[12] I. Koutis et al., *A nearly-m logn solver for SDD linear systems*, IEEE Symp. Foundations of Computer Science, 2011.
[13] D. Spielman et al., *Nearly-linear Time Algorithms for Graph Partitioning, Graph Sparsification, and Solving Linear Systems*, ACM Symp. Theory of Computing, 2004.
[14] G. Stewart, *Matrix Algorithms Volume II: Eigensystems*, SIAM, 2001.
[15] M. Kamon et al., *FastHenry: a multipole-accelerated 3-D inductance extraction program*, IEEE Trans. Microwave Theory and Techniques, vol. 42, pp. 1750-1758, 1994.
[16] S. Borm et al., *Introduction to Hierarchical Matrices with Applications*, Engineering Analysis with Boundary Elements, vol. 27, pp. 405-422, 2003.
[17] R. Kriemann, *Parallel $\mathcal{H}$-Matrix Arithmetics on Shared Memory Systems*, Computing vol. 74, pp. 273-297, 2005.
[18] R. Kriemann et al., *Parallel blackbox $\mathcal{H}$-LU preconditioning for elliptic boundary value problems*, Computing and Visualization in Sciences, vol. 11, pp. 273-291, 2007.