A Parallel Fast Transform-Based Preconditioning Approach for Electrical-Thermal Co-Simulation of Power Delivery Networks

Konstantis Daloukas, Alexia Marnari, Nestor Evmorfopoulos, Panagiota Tsompanopoulou, George. I. Stamoulis
Department of Computer and Communications Engineering,
University of Thessaly, Volos, Greece
{kodalouk, almarnar, nestevmo, yota, georges}@uth.gr

***Abstract*—Efficient analysis of massive on-chip power delivery networks is among the most challenging problems facing the EDA industry today. Due to Joule heating effect and the temperature dependence of resistivity, temperature is one of the most important factors that affect IR drop and must be taken into account in power grid analysis. However, the sheer size of modern power delivery networks (comprising several thousands or millions of nodes) usually forces designers to neglect thermal effects during IR drop analysis in order to simplify and accelerate simulation. As a result, the absence of accurate estimates of Joule heating effect on IR drop analysis introduces significant uncertainty in the evaluation of circuit functionality. This work presents a new approach for fast electrical-thermal co-simulation of large-scale power grids found in contemporary nanometer-scale ICs. A state-of-the-art iterative method is combined with an efficient and extremely parallel preconditioning mechanism, which enables harnessing the computational resources of massively parallel architectures, such as graphics processing units (GPUs). Experimental results demonstrate that the proposed method achieves a speedup of 66.1X for a 3.1M-node design over a state-of-the-art direct method and a speedup of 22.2X for a 20.9M-node design over a state-of-the-art iterative method when GPUs are utilized.**

I. Introduction

The relentless push for high-performance and low-power integrated circuits has been met by aggressive technology scaling, which enabled the integration of a vast number of devices on the same die but brought new problems and challenges to the surface. The on-chip power delivery network (power grid) constitutes a vital subsystem of modern nanometer-scale ICs, since it affects in a critical way the performance and correct functionality of the devices. Due to the aggressive scaling and the subsequent reduction in power supply voltage, IR drop in power delivery networks has become a very important factor that determines circuit reliability and performance. As a result, accurate analysis of power delivery networks is of great importance in order to ensure proper functionality.

The increased power density in conjunction with the lower power supply voltages due to shrinking sizes, has led to a significant increase in current density. Larger current densities result in greater IR drop, while Joule heating (self-heating) effect becomes of critical importance and should be seriously taken into account [3] as it contributes to temperature rise and affects reliability. As the electrical resistivity is temperature-dependent, temperature variations on the power delivery network substantially modify the interconnect resistances contributing to the IR drop in the power grid. However, such effects are usually neglected during IR drop analysis due to the immense growing size of modern power delivery networks and the corresponding demands in computational resources for their analysis. Power grids can be extremely large, demanding abundant computational resources, while at the same time, thermal analysis requires even more resources in terms of speed and memory.

As IR drop on the power grid and temperature rise on the interconnects constitute undesirable issues with respect to the performance, reliability and functionality of nano-scale technology, extensive research has been performed to separately analyze each of these issues. However, few approaches have been suggested for a thermal-aware IR drop analysis, despite the fact that IR drop and temperature are directly interdependent. Authors in [13] present an electrical-thermal co-simulation method, including Joule heating, air convection and fluidic cooling effects using the finite volume method with non-uniform rectangular grid. In the same context, a thermal-aware IR drop analysis for power grids is proposed in [15]. Both methods present the idea of iterative electrical-thermal simulations, updating the electrical resistivities in each iteration, including this way the thermal effects on each new electrical simulation. However, they focus on modeling aspects rather on simulation approaches.

Recently, parallel architectures have come to the forefront. Owing to their vast computational resources, they present a promising alternative for accelerating simulation algorithms. Approaches [9] and [8] present two methods for thermal and power delivery networks analysis respectively on GPUs. However, they do not consider a combined analysis approach and the proposed methods have limited potential for parallelism.

978-3-9815370-0-0/DATE13/©2013 EDAA

In this paper, we place specific emphasis on the simulation algorithms that are used in electrical-thermal co-simulation approaches. To that end, we propose a new algorithm for electrical-thermal co-simulation for large-scale power grids, found in most contemporary nano-scale ICs. Our method takes into account the basic thermal factors that contribute to temperature rises on the power grid such as Joule heating and heat from both the substrate and interconnects, while at the same time exhibits great potential of parallelism. We propose an efficient and highly-parallel preconditioning mechanism based on the application of a Fast Transform solver, which can be used in conjunction with a state-of-the-art iterative solution method in order to accelerate electrical-thermal analysis of power delivery networks. The benefits of the proposed method are twofold: i) the proposed preconditioning mechanism can accelerate the convergence rate of the iterative solution method by greatly reducing the required number of iterations, and ii) from a computational point of view, it exhibits near-optimal computational complexity, low memory requirements, and great potential for parallelism, which can harness the computational power of parallel architectures, such as multi-core processors or GPUs, thus further reducing the amount of time required for simulation. Experimental results demonstrate that our method achieves a speedup of 66.1X for a 3.1M-node design over a state-of-the-art direct method and a speedup of 22.2X for a 20.9M-node design over a state-of-the-art iterative method when GPUs are utilized. To the best of our knowledge, this is the first research approach that presents an algorithm for combined electrical-thermal simulation on parallel architectures.

The rest of the paper is organized as follows. We outline the modeling of power delivery networks and the corresponding thermal grid as well as the proposed combined simulation approach and the basic information about iterative solution methods in Section II. Section III presents the theory behind Fast Transform solvers for 2D and 3D networks. Next, we present the methodology behind the construction of efficient preconditioners for power and thermal grid analysis, as well as the proposed algorithm and the opportunities for parallelism in Section IV. We validate the performance of our approach using a series of large-scale designs in Section V. Finally, Section VI concludes the paper.

II. Overview of Methodology

A. Power Grid Electrical Modeling and Simulation

The typical model of power grid for transient analysis is obtained by modeling each wire segment (between two contacts) as a resistance in series (possibly) with an inductance, with capacitances to ground at both contact nodes. Assuming that the extracted electrical model is composed of N nodes, then by using the Modified Nodal Analysis (MNA) formulation and the Backward-Euler differential approximation, the transient simulation of the power grid entails the solution of the following $N \times N$ system of linear equations:

$$(\mathbf{G} + \frac{\mathbf{C}}{h_k})\mathbf{v}(h_k) = \frac{\mathbf{C}}{h_k}\mathbf{v}(h_{k-1}) + \mathbf{e}(h_k) \quad (1)$$

for each time-step $h_k,\ k = 1, \ldots$ (which may in general vary during the analysis). In the above, $\mathbf{G}$ is the $N \times N$ node conductance matrix, $\mathbf{C}$ is a $N \times N$ diagonal matrix of node capacitances, $\mathbf{v}(h_k)$ is the $N \times 1$ vector of node voltages, and $\mathbf{e}(h_k)$ is a $N \times 1$ vector of excitations from independent sources at the nodes. The incorporation of series inductances in the resistive branches rise to an analogous $N \times N$ recursive linear system and is described in [4].

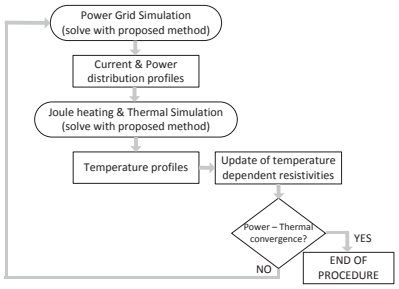


Fig. 1. Electrical-Thermal positive feedback simulation loop.

B. Power Grid Thermal Modeling and Simulation

Without loss of generality, in this work we focus on steady-state thermal analysis. Steady-state thermal analysis aims at determining the temperature distribution within a chip given a power density distribution that does not change with time. Steady-state analysis amounts to solving Poisson's equation:

$$q(r) = -k_t \nabla^2 T(r) \tag{2}$$

where r is the spatial coordinate of the point at which temperature is being determined, $q(r)$ is the rate of heat flow, T is the temperature, and k_t the thermal conductivity of the material.

By discretizing (2) using the Finite Difference Method (FDM), we obtain the following system of linear equations:

$$\mathbf{Gt} = \mathbf{p} \tag{3}$$

where $\mathbf{t}$ is the temperature vector at each point, $\mathbf{p}$ is the vector containing the total power generated within each element. Due to the electrical-thermal duality, each node in the discretization corresponds to a node in the circuit. Using the MNA formulation, matrix $\mathbf{G}$ contains the thermal resistors G_x, G_y, and G_z in the x, y, and z direction respectively, which can be computed as follows:

$$G_x = \frac{k_t(\Delta_y \cdot \Delta_z)}{\Delta_x},\ G_y = \frac{k_t(\Delta_x \cdot \Delta_z)}{\Delta_y},\ G_z = \frac{k_t(\Delta_x \cdot \Delta_y)}{\Delta_z}$$

where $\Delta_{\{x,y,z\}}$ is the length of the rectangle in each dimension.

C. Electro-Thermal Co-Simulation Approach

As was mentioned previously, due to temperature dependence of resistance and Joule self-heating of the conductors, the electrical and thermal characteristics of the power delivery network form a nonlinear system of equations. In order to capture the effect of the thermal profile of the power delivery network, we follow a combined electrical-thermal simulation for the power delivery network. Having set the initial input as well as the boundary conditions for the thermal simulation, the first step is the electrical analysis of the power grid. The power model provides the current and power distribution profiles to the thermal model. Subsequently, the thermal model estimates the temperature profile and once electrical resistivities have been updated, they are forwarded as a new input to the power model. After a number of iterations, power and temperature calculations converge and the thermal-aware power grid analysis is terminated. At each step where the solution of the power or the thermal grid (steps 1 and 2 respectively) are required, we apply the proposed methodology (the iterative linear solution method in conjunction with the proposed preconditioning mechanism) in order to accelerate the corresponding procedure. Fig. 1 depicts the flow diagram of the proposed approach.

D. Linear System Solution Methods

In the above procedure, steps 1 and 3 involve the solution of very large (and sparse) linear systems of equations $\mathbf{Ax} = \mathbf{b}$. Direct methods (based on matrix factorization) have been widely used in the past for solving the resulting linear systems, mainly because of their robustness in most types of problems. Unfortunately, these methods do not scale well with the dimension of the linear system, and become prohibitively expensive for large-scale networks and grids, in both execution time and memory requirements. Iterative methods involve only inner products and matrix-vector products, and constitute a better alternative for large sparse linear systems in many respects, being more computationally- and memory-efficient. System matrices arising from the modeling of the power and the thermal grid can also be shown to be Symmetric and Positive Definite (SPD), which allows the use of the extremely efficient method of Conjugate Gradient (CG) for the solution of the corresponding linear systems.

The main problem of iterative methods is their unpredictable rate of convergence which depends greatly on the properties (specifically the condition number) of the system matrix. A preconditioning mechanism, which transforms the linear system into one with more favorable properties, is essential to guarantee fast and robust convergence. However, the ideal preconditioner (one that approximates the system matrix well and is inexpensive to construct and apply) differs according to each particular problem and each different type of system matrix. That is why iterative methods have not reached the maturity of direct methods and have not yet gained widespread acceptance in linear circuit simulation. Although general-purpose preconditioners (such as incomplete factorizations or sparse approximate inverses) have been developed, they are not tuned to any particular simulation problems and cannot improve convergence by as much as specially-tailored preconditioners.

The preconditioning of iterative methods reduces to a preconditioner solve step $\mathbf{Mz} = \mathbf{r}$ in every iteration of the method, and effectively modifies the algorithm to solve the system $\mathbf{M}^{-1}\mathbf{Ax} = \mathbf{M}^{-1}\mathbf{b}$, which has the same solution as the original one $\mathbf{Ax} = \mathbf{b}$ [2]. If the preconditioner $\mathbf{M}$ approximates $\mathbf{A}$ in some way, then the condition number of the modified system $\mathbf{M}^{-1}\mathbf{Ax} = \mathbf{M}^{-1}\mathbf{b}$ becomes very small and the iterative method converges very quickly [2]. So the motivation behind preconditioning is to find a matrix $\mathbf{M}$ with the following properties: 1) the convergence rate of the preconditioned system $\mathbf{M}^{-1}\mathbf{Ax} = \mathbf{M}^{-1}\mathbf{b}$ is fast, and 2) a linear system involving $\mathbf{M}$ (i.e. $\mathbf{Mz} = \mathbf{r}$) - which then effectively receives the whole burden of the algorithm - is solved much more efficiently than the original system involving $\mathbf{A}$ (i.e. $\mathbf{Ax} = \mathbf{b}$). The effect of good preconditioning is even more pronounced if the operations for the solution of $\mathbf{Mz} = \mathbf{r}$ can be performed in parallel. Fortunately, as we will describe in Section IV, matrices arising from power grids and the electrical equivalent of thermal grids can be well-approximated by preconditioners with special structure such that the number of iterations is reduced to a great extent, while the systems $\mathbf{Mz} = \mathbf{r}$ can be solved by applying a Fast Transform in a near-optimal number of operations in a sequential implementation, and even less operations in a parallel environment (owing to the large parallel potential of Fast Transforms as well as other parallelization opportunities). The next section describes the special form of the preconditioner matrices, and the solution of the corresponding linear systems $\mathbf{Mz} = \mathbf{r}$ by Fast Transform solvers.

III. Fast Transform Solvers For Linear Systems With Special Structure

A. Fast Transform Solvers for 2D Networks

Let $\mathbf{M}$ be a $N \times N$ block-tridiagonal matrix with m blocks of size $n \times n$ each (overall $N = mn$) with the following form:

$$\mathbf{M} = \begin{bmatrix} \mathbf{T}_1 & -\gamma_1\mathbf{I}_n & & & \\ -\gamma_1\mathbf{I}_n & \mathbf{T}_2 & -\gamma_2\mathbf{I}_n & & \\ & \ddots & \ddots & \ddots & \\ & & -\gamma_{m-2}\mathbf{I}_n & \mathbf{T}_{m-1} & -\gamma_{m-1}\mathbf{I}_n \\ & & & -\gamma_{m-1}\mathbf{I}_n & \mathbf{T}_m \end{bmatrix} \tag{4}$$

where $\mathbf{I}_n$ is the $n \times n$ identity matrix and $\mathbf{T}_i$, $i = 1, \ldots, m$, are $n \times n$ tridiagonal matrices of the form:

$$\begin{aligned} \mathbf{T}_i &= \begin{bmatrix} \alpha_i + \beta_i & -\alpha_i & & & \\ -\alpha_i & 2\alpha_i + \beta_i & -\alpha_i & & \\ & \ddots & \ddots & \ddots & \\ & & -\alpha_i & 2\alpha_i + \beta_i & -\alpha_i \\ & & & -\alpha_i & \alpha_i + \beta_i \end{bmatrix} \\ &= \alpha_i \begin{bmatrix} 1 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 1 \end{bmatrix} + \beta_i \mathbf{I}_n \end{aligned} \tag{5}$$

We will describe an algorithm for the solution of a linear system $\mathbf{Mz} = \mathbf{r}$ with matrix $\mathbf{M}$ of the form (4), by the use of a Fast Transform solver in $\mathcal{O}(mn \log n) = \mathcal{O}(N \log n)$ operations. Such a solution is based on the fact that the eigen-decomposition of the tridiagonal matrices $\mathbf{T}_i$ is known beforehand, and that the matrices of eigenvectors that diagonalize $\mathbf{T}_i$ are matrices that correspond to a Fast Transform. More specifically, it can be shown [6] that each $\mathbf{T}_i$ has n distinct eigenvalues $\lambda_{i,j}$, $j = 1, \ldots, n$, which are given by:

$$\lambda_{i,j} = \beta_i + 4\alpha_i \sin^2 \frac{(j-1)\pi}{2n} = \beta_i + \alpha_i (2\cos\frac{(j-1)\pi}{n} - 2) \qquad (6)$$

as well as a set of n orthonormal eigenvectors $\mathbf{q}_j$, $j = 1, \ldots, n$, with elements:

$$q_{j,k} = \begin{cases} \sqrt{\frac{1}{n}} \cos\frac{(2k-1)(j-1)\pi}{2n} & j = 1, \quad k = 1, \ldots, n \\ \sqrt{\frac{2}{n}} \cos\frac{(2k-1)(j-1)\pi}{2n} & j = 2, \ldots, n, \quad k = 1, \ldots, n \end{cases} \qquad (7)$$

Note that the $\mathbf{q}_j$ do not depend on the values of α_i and β_i, and are the same for every matrix $\mathbf{T}_i$. If $\mathbf{Q}_n = [\mathbf{q}_1, \ldots, \mathbf{q}_n]$ denotes the matrix whose columns are the eigenvectors $\mathbf{q}_j$, then due to the eigen-decomposition of $\mathbf{T}_i$ we have $\mathbf{Q}_n^T \mathbf{T}_i \mathbf{Q}_n = \mathbf{\Lambda}_i = diag(\lambda_{i,1}, \ldots, \lambda_{i,n})$. By exploiting this diagonalization of the matrices $\mathbf{T}_i$, the system $\mathbf{M}\mathbf{z} = \mathbf{r}$ with $\mathbf{M}$ of the form (4) is equivalent to the following system (due to $\mathbf{Q}_n^T \mathbf{Q}_n = \mathbf{I}$):

$$\begin{bmatrix} \mathbf{Q}_n^T & & \\ & \ddots & \\ & & \mathbf{Q}_n^T \end{bmatrix} \mathbf{M} \begin{bmatrix} \mathbf{Q}_n & & \\ & \ddots & \\ & & \mathbf{Q}_n \end{bmatrix} \begin{bmatrix} \mathbf{Q}_n^T & & \\ & \ddots & \\ & & \mathbf{Q}_n^T \end{bmatrix} \mathbf{z} = \begin{bmatrix} \mathbf{Q}_n^T & & \\ & \ddots & \\ & & \mathbf{Q}_n^T \end{bmatrix} \mathbf{r}$$

$$\Leftrightarrow \begin{bmatrix} \mathbf{\Lambda}_1 & -\gamma_1\mathbf{I}_n & & & \\ -\gamma_1\mathbf{I}_n & \mathbf{\Lambda}_2 & -\gamma_2\mathbf{I}_n & & \\ & \ddots & \ddots & \ddots & \\ & & -\gamma_{m-2}\mathbf{I}_n & \mathbf{\Lambda}_{m-1} & -\gamma_{m-1}\mathbf{I}_n \\ & & & -\gamma_{m-1}\mathbf{I}_n & \mathbf{\Lambda}_m \end{bmatrix} \tilde{\mathbf{z}} = \tilde{\mathbf{r}} \qquad (8)$$

where

$$\tilde{\mathbf{z}} = \begin{bmatrix} \mathbf{Q}_n^T & & \\ & \ddots & \\ & & \mathbf{Q}_n^T \end{bmatrix} \mathbf{z}, \quad \tilde{\mathbf{r}} = \begin{bmatrix} \mathbf{Q}_n^T & & \\ & \ddots & \\ & & \mathbf{Q}_n^T \end{bmatrix} \mathbf{r}$$

If the $N \times 1$ vectors $\mathbf{r}$, $\mathbf{z}$, $\tilde{\mathbf{r}}$, $\tilde{\mathbf{z}}$ are also partitioned into m sub-vectors (blocks) of size $n \times 1$ each, i.e.

$$\mathbf{r} = \begin{bmatrix} \mathbf{r}_1 \\ \vdots \\ \mathbf{r}_m \end{bmatrix}, \quad \mathbf{z} = \begin{bmatrix} \mathbf{z}_1 \\ \vdots \\ \mathbf{z}_m \end{bmatrix}, \quad \tilde{\mathbf{r}} = \begin{bmatrix} \tilde{\mathbf{r}}_1 \\ \vdots \\ \tilde{\mathbf{r}}_m \end{bmatrix}, \quad \tilde{\mathbf{z}} = \begin{bmatrix} \tilde{\mathbf{z}}_1 \\ \vdots \\ \tilde{\mathbf{z}}_m \end{bmatrix}$$

then we have: $\tilde{\mathbf{r}}_i = \mathbf{Q}_n^T \mathbf{r}_i$ and $\tilde{\mathbf{z}}_i = \mathbf{Q}_n^T \mathbf{z}_i \Leftrightarrow \mathbf{z}_i = \mathbf{Q}_n \tilde{\mathbf{z}}_i$, $i = 1, \ldots, m$.

However, it can be shown [12] that each product $\mathbf{Q}_n^T \mathbf{r}_i = \tilde{\mathbf{r}}_i$ corresponds to a Discrete Cosine Transform of type-II (DCT-II) on $\mathbf{r}_i$, and each product $\mathbf{Q}_n \tilde{\mathbf{z}}_i = \mathbf{z}_i$ corresponds to an Inverse Discrete Cosine Transform of type-II (IDCT-II) on $\tilde{\mathbf{z}}_i$. This means that the computation of the whole vector $\tilde{\mathbf{r}}$ from $\mathbf{r}$ amounts to m independent DCT-II transforms of size n, and the computation of the whole vector $\mathbf{z}$ from $\tilde{\mathbf{z}}$ amounts to m independent IDCT-II transforms of size n. A modification of Fast Fourier Transform (FFT) can be employed for each of the m independent DCT-II/IDCT-II transforms [12], giving a total operation count of $\mathcal{O}(mn \log n) = \mathcal{O}(N \log n)$.

If now $\mathbf{P}$ is a permutation matrix that reorders the elements of a vector or the rows of a matrix as $1, n+1, 2n+1, \ldots, (m-1)n+1,\ 2, n+2, 2n+2, \ldots, (m-1)n+2,\ \ldots,\ n, n+n, 2n+n, \ldots, (m-1)n+n$, and $\mathbf{P}^T$ is the inverse permutation matrix, then the system (8) is further equivalent to:

$$\mathbf{P} \begin{bmatrix} \mathbf{\Lambda}_1 & -\gamma_1\mathbf{I}_n & & & \\ -\gamma_1\mathbf{I}_n & \mathbf{\Lambda}_2 & -\gamma_2\mathbf{I}_n & & \\ & \ddots & \ddots & \ddots & \\ & & -\gamma_{m-2}\mathbf{I}_n & \mathbf{\Lambda}_{m-1} & -\gamma_{m-1}\mathbf{I}_n \\ & & & -\gamma_{m-1}\mathbf{I}_n & \mathbf{\Lambda}_m \end{bmatrix} \mathbf{P}^T \mathbf{P}\tilde{\mathbf{z}} = \mathbf{P}\tilde{\mathbf{r}}$$

$$\Leftrightarrow \begin{bmatrix} \tilde{\mathbf{T}}_1 & & \\ & \ddots & \\ & & \tilde{\mathbf{T}}_n \end{bmatrix} \tilde{\mathbf{z}}^P = \tilde{\mathbf{r}}^P \qquad (9)$$

where

$$\tilde{\mathbf{T}}_j = \begin{bmatrix} \lambda_{1,j} & -\gamma_1 & & & \\ -\gamma_1 & \lambda_{2,j} & -\gamma_2 & & \\ & \ddots & \ddots & \ddots & \\ & & -\gamma_{m-2} & \lambda_{m-1,j} & -\gamma_{m-1} \\ & & & -\gamma_{m-1} & \lambda_{m,j} \end{bmatrix} \qquad (10)$$

and $\tilde{\mathbf{z}}^P = \mathbf{P}\tilde{\mathbf{z}}$, $\tilde{\mathbf{r}}^P = \mathbf{P}\tilde{\mathbf{r}}$. If the $N \times 1$ vectors $\tilde{\mathbf{z}}^P$, $\tilde{\mathbf{r}}^P$ are partitioned into n sub-vectors $\tilde{\mathbf{z}}_j^P, \tilde{\mathbf{r}}_j^P$ of size $m \times 1$ each, then the system (9) effectively represents n independent tridiagonal systems $\tilde{\mathbf{T}}_j \tilde{\mathbf{z}}_j^P = \tilde{\mathbf{r}}_j^P$ of size m which can be solved w.r.t. the blocks $\tilde{\mathbf{z}}_j^P$, $j = 1, \ldots, n$ (to produce the whole vector $\tilde{\mathbf{z}}^P$) in a total of $\mathcal{O}(mn) = \mathcal{O}(N)$ operations. For each such system the coefficient matrix (10) is known beforehand and is determined exclusively by the eigenvalues (6) and the values γ_i of matrix $\mathbf{M}$, while the right-hand side (RHS) vector $\tilde{\mathbf{r}}_j^P$ is composed of specific components of (DCT-II)-transformed blocks of vector $\mathbf{r}$. The equivalence of the system $\mathbf{M}\mathbf{z} = \mathbf{r}$, with $\mathbf{M}$ as in (4), to the system (9) gives a procedure for fast solution of $\mathbf{M}\mathbf{z} = \mathbf{r}$, which is described in algorithm from Fig. 2. Note that apart from the resultant near-optimal complexity of $\mathcal{O}(N \log n)$ operations, the m DCT-II/IDCT-II transforms and the n tridiagonal systems are completely independent to each other and can be solved in parallel. Furthermore, both the FFT and the tridiagonal solution algorithm are highly-parallel algorithms by themselves, allowing for further acceleration of the individual transforms/solutions when executed on parallel platforms.

B. Fast Transform Solvers for 3D Networks

Let $\mathbf{M}$ be a $N \times N$ block-tridiagonal matrix with l blocks of size $mn \times mn$ each (overall $N = lmn$) with the following form:

$$\mathbf{M} = \begin{bmatrix} \mathbf{M}_1 & -\delta_1\mathbf{I}_{mn} & & & \\ -\delta_1\mathbf{I}_{mn} & \mathbf{M}_2 & -\delta_2\mathbf{I}_{mn} & & \\ & \ddots & \ddots & \ddots & \\ & & -\delta_{l-2}\mathbf{I}_{mn} & \mathbf{M}_{l-1} & -\delta_{l-1}\mathbf{I}_{mn} \\ & & & -\delta_{l-1}\mathbf{I}_{mn} & \mathbf{M}_l \end{bmatrix} \qquad (11)$$

where $\mathbf{I}_{mn}$ is the $mn \times mn$ identity matrix, $\mathbf{M}_i$, $i = 1, \ldots, l$, are themselves $mn \times mn$ block-tridiagonal matrices of the form:

$$\mathbf{M}_i = \begin{bmatrix} \mathbf{T}_i + \gamma_i\mathbf{I}_n & -\gamma_i\mathbf{I}_n & & & \\ -\gamma_i\mathbf{I}_n & \mathbf{T}_i + 2\gamma_i\mathbf{I}_n & -\gamma_i\mathbf{I}_n & & \\ & \ddots & \ddots & \ddots & \\ & & -\gamma_i\mathbf{I}_n & \mathbf{T}_i + 2\gamma_i\mathbf{I}_n & -\gamma_i\mathbf{I}_n \\ & & & -\gamma_i\mathbf{I}_n & \mathbf{T}_i + \gamma_i\mathbf{I}_n \end{bmatrix}$$

where $\mathbf{I}_n$ is the $n \times n$ identity matrix and $\mathbf{T}_i$ have the form (10). Thus, the eigenvectors of the diagonal blocks of $\mathbf{M}_i$ are the same as those of $\mathbf{T}_i$, with values given by (7). By a similar reasoning as in (8), the linear system $\mathbf{M}\mathbf{z} = \mathbf{r}$ with $\mathbf{M}$ of the form (11) is equivalent to the following:

$$\begin{bmatrix} \tilde{\mathbf{M}}_1 & -\delta_1\mathbf{I}_{mn} & & & \\ -\delta_1\mathbf{I}_{mn} & \tilde{\mathbf{M}}_2 & -\delta_2\mathbf{I}_{mn} & & \\ & \ddots & \ddots & \ddots & \\ & & -\delta_{l-2}\mathbf{I}_{mn} & \tilde{\mathbf{M}}_{l-1} & -\delta_{l-1}\mathbf{I}_{mn} \\ & & & -\delta_{l-1}\mathbf{I}_{mn} & \tilde{\mathbf{M}}_l \end{bmatrix} \tilde{\mathbf{z}} = \tilde{\mathbf{r}} \qquad (12)$$

where

$$\tilde{\mathbf{M}}_i = \begin{bmatrix} \mathbf{\Lambda}_i^{(1)} & -\gamma_i\mathbf{I}_n & & & \\ -\gamma_i\mathbf{I}_n & \mathbf{\Lambda}_i^{(2)} & -\gamma_i\mathbf{I}_n & & \\ & \ddots & \ddots & \ddots & \\ & & -\gamma_i\mathbf{I}_n & \mathbf{\Lambda}_i^{(2)} & -\gamma_i\mathbf{I}_n \\ & & & -\gamma_i\mathbf{I}_n & \mathbf{\Lambda}_i^{(1)} \end{bmatrix}$$

$$\tilde{\mathbf{z}} = \begin{bmatrix} \mathbf{Q}_n^T & & \\ & \ddots & \\ & & \mathbf{Q}_n^T \end{bmatrix} \mathbf{z}, \quad \tilde{\mathbf{r}} = \begin{bmatrix} \mathbf{Q}_n^T & & \\ & \ddots & \\ & & \mathbf{Q}_n^T \end{bmatrix} \mathbf{r},$$

Fig. 2. Fast Transform algorithm for the preconditioner solve step $\mathbf{M}\mathbf{z} = \mathbf{r}$ in 2D networks

1: Partition the RHS vector $\mathbf{r}$ into m blocks $\mathbf{r}_i$ of size n, and perform DCT-II transform ($\mathbf{Q}_n^T \mathbf{r}_i$) on each block to obtain transformed vector $\tilde{\mathbf{r}}$
2: Permute vector $\tilde{\mathbf{r}}$ by permutation $\mathbf{P}$, which orders elements as $1, n+1, \ldots, (m-1)n+1,\ 2, n+2, \ldots, (m-1)n+2,\ \ldots,\ n, n+n, \ldots, (m-1)n+n$, in order to obtain vector $\tilde{\mathbf{r}}^P$
3: Solve the n tridiagonal systems (9) with known coefficient matrices (10), in order to obtain vector $\tilde{\mathbf{z}}^P$.
4: Apply inverse permutation $\mathbf{P}^T$ on vector $\tilde{\mathbf{z}}^P$ so as to obtain vector $\tilde{\mathbf{z}}$.
5: Partition vector $\tilde{\mathbf{z}}$ into m blocks $\tilde{\mathbf{z}}_i$ of size n, and perform IDCT-II transform ($\mathbf{Q}_n \tilde{\mathbf{z}}_i$) on each block to obtain final solution vector $\mathbf{z}$

and $\mathbf{\Lambda}_i^{(1)} = diag(\lambda_{i,1}^{(1)}, \ldots, \lambda_{i,n}^{(1)})$, $\mathbf{\Lambda}_i^{(2)} = diag(\lambda_{i,1}^{(2)}, \ldots, \lambda_{i,n}^{(2)})$ are diagonal matrices with the eigenvalues of $\mathbf{T}_i + \gamma_i \mathbf{I}_n$, $\mathbf{T}_i + 2\gamma_i \mathbf{I}_n$, which are the following:

$$\lambda_{i,j}^{(1)} = \gamma_i + \beta_i + \alpha_i(2\cos\frac{(j-1)\pi}{n} - 2),\ j = 1, \ldots, n$$
$$\lambda_{i,j}^{(2)} = 2\gamma_i + \beta_i + \alpha_i(2\cos\frac{(j-1)\pi}{n} - 2),\ j = 1, \ldots, n$$

If $\mathbf{P}$ is again the $mn \times mn$ permutation matrix that reorders the elements of a vector or the rows of a matrix as $1, n+1, \ldots, (m-1)n+1,\ 2, n+2, \ldots, (m-1)n+2,\ \ldots,\ n, n+n, \ldots, (m-1)n+n$, and $\mathbf{P_1}$, $\mathbf{P_1^T}$ denote the block-diagonal $lmn \times lmn$ permutation matrices $\mathbf{P_1} = diag(\mathbf{P}, \ldots, \mathbf{P})$, $\mathbf{P_1^T} = diag(\mathbf{P}^T, \ldots, \mathbf{P}^T)$, then the system (12) is further equivalent to:

$$\begin{bmatrix} \mathbf{D}_1 & -\delta_1\mathbf{I}_{mn} & & & \\ -\delta_1\mathbf{I}_{mn} & \mathbf{D}_2 & -\delta_2\mathbf{I}_{mn} & & \\ & \ddots & \ddots & \ddots & \\ & & -\delta_{l-2}\mathbf{I}_{mn} & \mathbf{D}_{l-1} & -\delta_{l-1}\mathbf{I}_{mn} \\ & & & -\delta_{l-1}\mathbf{I}_{mn} & \mathbf{D}_l \end{bmatrix} \tilde{\mathbf{z}}^{\mathbf{P_1}} = \tilde{\mathbf{r}}^{\mathbf{P_1}} \tag{13}$$

where $\mathbf{D}_i = diag(\tilde{\mathbf{T}}_{i,1}, \ldots, \tilde{\mathbf{T}}_{i,n})$, $i = 1, \ldots, l$, with $\tilde{\mathbf{T}}_{i,j}$, $j = 1, \ldots, n$ being $m \times m$ tridiagonal matrices of the form:

$$\tilde{\mathbf{T}}_{i,j} = \begin{bmatrix} \lambda_{i,j}^{(1)} & -\gamma_i & & & \\ -\gamma_i & \lambda_{i,j}^{(2)} & -\gamma_i & & \\ & \ddots & \ddots & \ddots & \\ & & -\gamma_i & \lambda_{i,j}^{(2)} & -\gamma_i \\ & & & -\gamma_i & \lambda_{i,j}^{(1)} \end{bmatrix}$$
$$= \gamma_i \begin{bmatrix} 1 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 1 \end{bmatrix} + (\alpha_i(2\cos\frac{(j-1)\pi}{n} - 2) + \beta_i)\mathbf{I}_m$$

and $\tilde{\mathbf{z}}^{\mathbf{P_1}} = \mathbf{P_1}\tilde{\mathbf{z}}, \tilde{\mathbf{r}}^{\mathbf{P_1}} = \mathbf{P_1}\tilde{\mathbf{r}}$. If $\tilde{\mathbf{\Lambda}}_{i,j} = diag(\tilde{\lambda}_{i,j,1}, \ldots, \tilde{\lambda}_{i,j,m})$ is the diagonal matrix with the eigenvalues of $\tilde{\mathbf{T}}_{i,j}$, which are:

$$\tilde{\lambda}_{i,j,k} = \gamma_i(2\cos\frac{(k-1)\pi}{m} - 2) + \alpha_i(2\cos\frac{(j-1)\pi}{n} - 2) + \beta_i,\ k = 1, \ldots, m \tag{14}$$

and $\mathbf{Q}_m$ is the common matrix of eigenvectors for all $\tilde{\mathbf{T}}_{i,j}$, then again by similar reasoning as in (8), the system (13) is equivalent to:

$$\begin{bmatrix} \tilde{\mathbf{D}}_1 & -\delta_1\mathbf{I}_{mn} & & & \\ -\delta_1\mathbf{I}_{mn} & \tilde{\mathbf{D}}_2 & -\delta_2\mathbf{I}_{mn} & & \\ & \ddots & \ddots & \ddots & \\ & & -\delta_{l-2}\mathbf{I}_{mn} & \tilde{\mathbf{D}}_{l-1} & -\delta_{l-1}\mathbf{I}_{mn} \\ & & & -\delta_{l-1}\mathbf{I}_{mn} & \tilde{\mathbf{D}}_l \end{bmatrix} \tilde{\tilde{\mathbf{z}}} = \tilde{\tilde{\mathbf{r}}} \tag{15}$$

where $\tilde{\mathbf{D}}_i = diag(\tilde{\mathbf{\Lambda}}_{i,1}, \ldots, \tilde{\mathbf{\Lambda}}_{i,n})$ and

$$\tilde{\tilde{\mathbf{z}}} = \begin{bmatrix} \mathbf{Q}_m^T & & \\ & \ddots & \\ & & \mathbf{Q}_m^T \end{bmatrix} \tilde{\mathbf{z}}^{\mathbf{P_1}}, \quad \tilde{\tilde{\mathbf{r}}} = \begin{bmatrix} \mathbf{Q}_m^T & & \\ & \ddots & \\ & & \mathbf{Q}_m^T \end{bmatrix} \tilde{\mathbf{r}}^{\mathbf{P_1}}$$

If now $\mathbf{P}_2$ is a permutation matrix of size $N \times N$ that reorders the elements of a vector or the rows of a matrix as $1, mn+1, 2mn+1, \ldots, (l-1)mn+1,\ \ 2, mn+2, 2mn+2, \ldots, (l-1)mn+2,\ \ldots,\ \ mn, mn+mn, 2mn+mn, \ldots, (l-1)mn+mn$, and $\mathbf{P}_2^T$ is the inverse permutation matrix, then system (15) is equivalent to:

$$\tilde{\tilde{\mathbf{M}}}\tilde{\tilde{\mathbf{z}}}^{\mathbf{P_2}} = \tilde{\tilde{\mathbf{r}}}^{\mathbf{P_2}} \tag{16}$$

where $\tilde{\tilde{\mathbf{M}}} = diag(\tilde{\tilde{\mathbf{T}}}_{1,1}, \tilde{\tilde{\mathbf{T}}}_{1,2} \ldots, \tilde{\tilde{\mathbf{T}}}_{1,m}, \tilde{\tilde{\mathbf{T}}}_{2,1}, \ldots, \tilde{\tilde{\mathbf{T}}}_{2,m}, \ldots, \tilde{\tilde{\mathbf{T}}}_{n,m})$, with $\tilde{\tilde{\mathbf{T}}}_{j,k}$, $j = 1, \ldots, n$, $k = 1, \ldots, m$ being $l \times l$ tridiagonal matrices of the form:

$$\tilde{\tilde{\mathbf{T}}}_{j,k} = \begin{bmatrix} \tilde{\lambda}_{1,j,k} & -\delta_1 & & & \\ -\delta_1 & \tilde{\lambda}_{2,j,k} & -\delta_2 & & \\ & \ddots & \ddots & \ddots & \\ & & -\delta_{l-1} & \tilde{\lambda}_{l-1,j,k} & -\delta_l \\ & & & -\delta_l & \tilde{\lambda}_{l,j,k} \end{bmatrix} \tag{17}$$

and $\tilde{\tilde{\mathbf{z}}}^{\mathbf{P_2}} = \mathbf{P}_2\tilde{\tilde{\mathbf{z}}}, \tilde{\tilde{\mathbf{r}}}^{\mathbf{P_2}} = \mathbf{P}_2\tilde{\tilde{\mathbf{r}}}$. The equivalence of the system $\mathbf{Mz} = \mathbf{r}$, with $\mathbf{M}$ as in (11), to the system (16), gives a procedure for fast solution of $\mathbf{Mz} = \mathbf{r}$ which is described in algorithm in Fig. 3.

IV. Proposed Approach for Electro-Thermal Simulation

As mentioned in Section II-D, the intuition behind preconditioner's formulation is to create a matrix $\mathbf{M}$ that will approximate the system matrix $\mathbf{A}$ as faithfully as possible, while at the same time enable the utilization of efficient algorithms for the solution of systems $\mathbf{Mz} = \mathbf{r}$. We have developed such an algorithm based on a Fast Transform solver in the previous section for matrices with special structure arising from 2D and 3D problems such as the power and the thermal grid. This section will describe the construction of the preconditioners that are utilized in the Preconditioned Conjugate Gradient (PCG) method for the simulation of the power and thermal grid.

A. Power Grid Preconditioner Construction

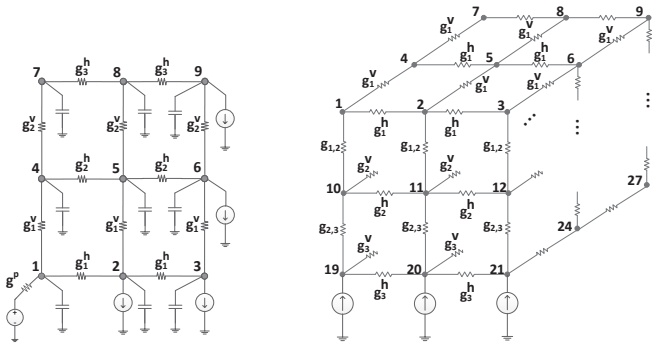
Practical power grids are created as orthogonal wire meshes with very regular spatial geometries, with possibly some irregularities imposed by design constraints (e.g. some missing connections between adjacent nodes), and arranged in a few - typically 2 to 6 - metal layers of alternating routing directions (horizontal and vertical). Due to the presence of vias between successive metal layers, the actual grid has the structure of a 3D mesh, with very few planes along the third dimension. However, as it was observed in [8], electrical resistances of vias are usually much smaller than wire resistances, leading to voltage drops much less than 1mV. Also, data in [11] show that almost all circuit elements (mainly resistances) in each metal layer have the same values (with few differences due to grid irregularities).

Based on the above observations, we create a preconditioner matrix that approximates the system matrix of the power grid by a process of regularization of the 3D power grid to a regular 2D grid, consisting of the following steps:

1) Determine the distinct x- and y-coordinates of all nodes in the different layers of the 3D grid, and take their Cartesian product to specify the location of the nodes in the regular 2D grid.
2) By disregarding via resistances between layers, collapse the 3D grid onto the regular 2D grid by adding together all horizontal branch conductances $g^h \equiv \frac{1}{r^h}$ connected in parallel between adjacent nodes in the x-direction of the 2D grid, and all vertical branch conductances $g^v \equiv \frac{1}{r^v}$ connected in parallel between adjacent nodes in the y-direction of the 2D grid (where r^h and r^v denote the resistance of horizontal and vertical branches). If a conductance of the 3D grid occupies multiple nodes of the regular 2D grid, it is decomposed into a corresponding number of pieces. The node capacitances corresponding to the same regular grid nodes are also added together during the collapsing.
3) In the regular 2D grid, substitute horizontal branch conductances by their average value in each horizontal rail, and vertical branch conductances by their average value in each horizontal slice (enclosed between two adjacent horizontal rails). Substitute node capacitances in each horizontal rail by their average value as well.

Fig. 3. Fast Transform algorithm for the preconditioner solve step $\mathbf{Mz} = \mathbf{r}$ in 3D Networks

1: Partition the RHS vector $\mathbf{r}$ into lm sub-vectors $\mathbf{r}_i$ of size n, and perform DCT-II transform ($\mathbf{Q}_n^T\mathbf{r}_i$) on each sub-vector to obtain transformed vector $\tilde{\mathbf{r}}$.
2: Partition vector $\tilde{\mathbf{r}}$ into l sub-vectors $\tilde{\mathbf{r}}_i$ of size mn, and permute each sub-vector by permutation $\mathbf{P}$, which orders elements as $1, n+1, \ldots, (m-1)n+1,\ 2, n+2, \ldots, (m-1)n+2,\ \ldots,\ n, n+n, \ldots, (m-1)n+n$, in order to obtain vector $\tilde{\mathbf{r}}^{\mathbf{P_1}}$.
3: Partition vector $\tilde{\mathbf{r}}^{\mathbf{P_1}}$ into ln sub-vectors $\tilde{\mathbf{r}}_i^{\mathbf{P_1}}$ of size m, and perform DCT-II transform ($\mathbf{Q}_m^T\tilde{\mathbf{r}}_i^{\mathbf{P_1}}$) on each sub-vector to obtain transformed vector $\tilde{\tilde{\mathbf{r}}}$.
4: Permute vector $\tilde{\tilde{\mathbf{r}}}$ by applying permutation $\mathbf{P}_2$, which orders elements as $1, mn+1, 2mn+1, \ldots, (l-1)mn+1,\ 2, mn+2, 2mn+2, \ldots, (l-1)mn+2,\ \ldots,\ mn, mn+mn, 2mn+mn, \ldots, (l-1)mn+mn$, in order to obtain vector $\tilde{\tilde{\mathbf{r}}}^{P_2}$.
5: Solve the mn tridiagonal systems (16) with known coefficient matrices (17), in order to obtain vector $\tilde{\tilde{\mathbf{z}}}^{P_2}$
6: Apply inverse permutation $\mathbf{P}_2^T$ on vector $\tilde{\tilde{\mathbf{z}}}^{P_2}$ so as to obtain vector $\tilde{\tilde{\mathbf{z}}}$.
7: Partition vector $\tilde{\tilde{\mathbf{z}}}$ into ln sub-vectors $\tilde{\tilde{\mathbf{z}}}_i$ of size m, and perform IDCT-II transform ($\mathbf{Q}_m\tilde{\tilde{\mathbf{z}}}_i$) on each sub-vector to obtain vector $\tilde{\mathbf{z}}^{\mathbf{P_1}}$.
8: Partition vector $\tilde{\mathbf{z}}^{\mathbf{P_1}}$ into l sub-vectors $\tilde{\mathbf{z}}_i^{\mathbf{P_1}}$ of size mn, and apply inverse permutation $\mathbf{P}_1^T$ on each sub-vector to obtain vector $\tilde{\mathbf{z}}$.
9: Partition vector $\tilde{\mathbf{z}}$ into lm sub-vectors $\tilde{\mathbf{z}}_i$ of size n, and perform IDCT-II transform ($\mathbf{Q}_n\tilde{\mathbf{z}}_i$) on each sub-vector to obtain final solution vector $\mathbf{z}$.



(a) Example of a regular 2D power grid obtained after the regularization process.

(b) Example of a 3D thermal grid that is used for preconditioning.

Fig. 4. Examples of a power delivery and a thermal grid that are used for preconditioning.

Fig. 4(a) shows an example of a 2D regular grid that results from the previous regularization process used to construct the preconditioner matrix. If we use the depicted natural node numbering (proceeding horizontally, since this is always the routing direction of the lowest-level metal layer), the matrix $\mathbf{G} + \frac{\mathbf{C}}{h_k}$ that corresponds to the regular 2D grid will be the following block-tridiagonal matrix:

$$\mathbf{M} = \begin{bmatrix} \mathbf{T}_1 & -g_1^v\mathbf{I} & \\ -g_1^v\mathbf{I} & \mathbf{T}_2 & -g_2^v\mathbf{I} \\ & -g_2^v\mathbf{I} & \mathbf{T}_3 \end{bmatrix}$$

where $\mathbf{T}_1, \mathbf{T}_2, \mathbf{T}_3$ are 3×3 tridiagonal matrices (each one corresponding to a horizontal rail of the 2D grid) which have the form:

$$\mathbf{T}_1 = \begin{bmatrix} g_1^h+g_1^v+g^p+\frac{c_1}{h_k} & -g_1^h & \\ -g_1^h & 2g_1^h+g_1^v+\frac{c_1}{h_k} & -g_1^h \\ & -g_1^h & g_1^h+g_1^v+\frac{c_1}{h_k} \end{bmatrix}$$

$$\mathbf{T}_2 = \begin{bmatrix} g_2^h+g_1^v+g_2^v+\frac{c_2}{h_k} & -g_2^h & \\ -g_2^h & 2g_2^h+g_1^v+g_2^v+\frac{c_2}{h_k} & -g_2^h \\ & -g_2^h & g_2^h+g_1^v+g_2^v+\frac{c_2}{h_k} \end{bmatrix}$$

$$\mathbf{T}_3 = \begin{bmatrix} g_3^h+g_2^v+g_3^v+\frac{c_3}{h_k} & -g_3^h & \\ -g_3^h & 2g_3^h+g_2^v+g_3^v+\frac{c_3}{h_k} & -g_3^h \\ & -g_3^h & g_3^h+g_2^v+g_3^v+\frac{c_3}{h_k} \end{bmatrix}$$

In the above, g_i^h is the average horizontal conductance in the i-th horizontal rail, g_i^v is the average vertical conductance in the i-th horizontal slice, and c_i is the average node capacitance in the i-th horizontal rail. Also h_k is the current analysis time-step (possibly variable), and $g^p \equiv \frac{1}{r^p}$ is the parasitic conductance of the supply pads.

We observe that the form of the above matrix is almost identical to (4), with the exception of the pad parasitic conductance g^p in few places along the diagonal (considering that the number of voltage pads is much smaller than the number of nodes N). In order to obtain a preconditioner $\mathbf{M}$ with an exact form that can be efficiently solved by the application of a Fast Transform, we can just omit entirely those pad parasitics. However, we have found that in practice it is usually better to amortize the total sum of pad conductances of a specific horizontal rail (in the regular 2D grid) to all nodes of this rail, i.e. assume that all nodes of the i-th horizontal rail have pad conductance $\bar{g}_i^p = \frac{(\sum g^p)_i}{n}$, where $\left(\sum g^p\right)_i$ is the sum of the actual pad conductances attached to nodes of the i-th horizontal rail. This also has the beneficial effect of making the preconditioner $\mathbf{M}$ non-singular in the case of DC analysis (where capacitances are absent). In the above example, the block $\mathbf{T}_1$ would become:

$$\mathbf{T}_1 = \begin{bmatrix} g_1^h+g_1^v+\bar{g}_1^p+\frac{c_1}{h_k} & -g_1^h & \\ -g_1^h & 2g_1^h+g_1^v+\bar{g}_1^p+\frac{c_1}{h_k} & -g_1^h \\ & -g_1^h & g_1^h+g_1^v+\bar{g}_1^p+\frac{c_1}{h_k} \end{bmatrix}$$

where $\bar{g}_1^p = \frac{g^p}{3}$. It is not difficult to generalize the procedure to an arbitrary $m \times n$ power grid. In that case, the preconditioner will comprise m blocks of size $n \times n$ and have the form (4), where $\alpha_i = g_i^h,\ \beta_i = g_i^v + g_{i-1}^v + \bar{g}_i^p + \frac{c_i}{h_k},\ \gamma_i = g_i^v, \quad i = 1, \ldots, m$ (with $g_0^v = g_m^v = 0$).

B. Thermal Grid Preconditioner Construction

Typically, to model the thermal profile of the power grid, a chip is considered as comprising n layers, where each layer contains metal lines and inter-layer insulator. The topmost layer is covered by a thermal insulation layer and the heat generated in the power grid is conducted away by the substrate (usually attached to a heat sink). By modeling each layer as was mentioned in section II-B, the thermal grid is equivalent to a highly regular resistive network, with resistive branches connecting nodes in the *x*, *y*, and *z* axis. To create a preconditioner that will approximate the grid matrix, we substitute each horizontal and vertical thermal conductance with its average value in the corresponding layer. Moreover, we substitute each thermal conductance $g_{i,i+1}$ connecting nodes in adjacent layers (z axis) with their average value between the two layers. Fig. 4(b) is an example of a thermal grid with $n = 3$, $m = 3$ nodes in the *x* and *y* axis respectively and $l = 3$ layers. Using the depicted numbering, the matrix that corresponds to the aforementioned grid is the following block-tridiagonal matrix:

$$\mathbf{M} = \begin{bmatrix} \mathbf{M}_1 & -g_{1,2}\mathbf{I}_{mn} & \\ -g_{1,2}\mathbf{I}_{mn} & \mathbf{M}_2 & -g_{2,3}\mathbf{I}_{mn} \\ & -g_{2,3}\mathbf{I}_{mn} & \mathbf{M}_3 \end{bmatrix}$$

where

$$\mathbf{M}_i = \begin{bmatrix} \mathbf{T}_i + g_i^v\mathbf{I}_n & -g_i^v\mathbf{I}_n & \\ -g_i^v\mathbf{I}_n & \mathbf{T}_i + 2g_i^v\mathbf{I}_n & -g_i^v\mathbf{I}_n \\ & -g_i^v\mathbf{I}_n & \mathbf{T}_i + g_i^v\mathbf{I}_n \end{bmatrix}, \quad i = 1, 2, 3$$

$$\mathbf{T}_1 = \begin{bmatrix} g_1^h + g_{1,2} & -g_1^h & \\ -g_1^h & 2g_1^h + g_{1,2} & -g_1^h \\ & -g_1^h & g_2^h + g_{1,2} \end{bmatrix}$$

$$\mathbf{T}_2 = \begin{bmatrix} g_2^h + g_{1,2} + g_{2,3} & -g_2^h & \\ -g_2^h & 2g_2^h + g_{1,2} + g_{2,3} & -g_2^h \\ & -g_2^h & g_2^h + g_{1,2} + g_{2,3} \end{bmatrix}$$

$$\mathbf{T}_3 = \begin{bmatrix} g_3^h + g_{2,3} & -g_3^h & \\ -g_3^h & 2g_3^h + g_{2,3} & -g_3^h \\ & -g_3^h & g_3^h + g_{2,3} \end{bmatrix}$$

Similarly to the power grid preconditioner, the form of the above matrix is identical to matrix (11). As a result, if such a matrix is used as the preconditioner $\mathbf{M}$ for the thermal analysis procedure, it can be efficiently solved through utilization of a Fast Transform solver, as was described in section III-B.

C. Procedure Implementation

Once the preconditioners for the power and the thermal grid have been created, the proposed approach executes the electro-thermal loop in Fig. 1. At each step this requires the solution of the power grid at step 1 and the solution of the thermal grid at step 3. An off-the-self implementation of the PCG method can be used for both steps, with an external call for the preconditioner-solve step $\mathbf{Mz} = \mathbf{r}$. The latter corresponds to algorithm in Fig. 2 for power grid electrical simulation and algorithm in Fig. 3 for power grid thermal simulation.

Owing to their special construction that allow utilization of Fast Transform solvers, solution of the preconditioned systems of the power and the thermal grid offer ample parallelism, both at data- and task-level. Both FFT and the tridiagonal system solution are highly-parallel algorithms that offer abundant level of data-level parallelism [7] [14]. Thus, the proposed method can efficiently utilize the computational resources found in massively parallel architectures, thus greatly accelerating the simulation process. This comes in contrast with most widely-used preconditioning methods, such as incomplete factorizations, which have limited parallelism.

As far as task-level parallelism is concerned, algorithms from Fig. 2 and Fig. 3 entail a number of independent one-dimensional DCT-II and IDCT-II transforms as well as the solution of a large number of independent tridiagonal systems. This translates to additional task-level parallelism, which can result to further acceleration on *multi*-GPU systems for the preconditioner solve step as the independent transforms and tridiagonal solvers can be executed in parallel, requiring limited communication between different GPUs.

TABLE I

RUNTIME RESULTS FOR THE THREE SOLVERS. $Iter.$ IS THE AVERAGE NUMBER OF ITERATIONS (TOTAL NUMBER OF ITERATIONS IN EACH ITERATION OVER THE NUMBER OF ITERATIONS REQUIRED FOR CONVERGENCE OF THE ELECTRO-THERMAL LOOP) REQUIRED FOR CONVERGENCE OF EACH ITERATIVE METHOD. $Time$ DENOTES THE AVERAGE TIME REQUIRED FOR THE SOLUTION AT EACH ITERATION, WHILE Spd_{CHOL} AND Spd_{ICCG} DENOTE THE SPEEDUP OF ET-FTCG OVER CHOLMOD AND ICCG RESPECTIVELY. THE CONVERGENCE TOLERANCE FOR ITERATIVE SOLVERS WAS 10^{-6} AND CONVERGENCE WAS ACHIEVED IN ALL CASES.

Benchmark		CHOLMOD	ICCG		ET-FTCG			
	$Nodes$	$Time\ (s)$	$Iter.$	$Time\ (s)$	$Iter.$	$Time\ (s)$	Spd_{CHOL}	Spd_{ICCG}
ckt1	3.1M	105.8	201	15.1	62	1.6	66.1X	9.4X
ckt2	6.3M	N/A	296	58.1	63	3.7	N/A	15.7X
ckt3	14.6M	N/A	465	214.1	62	10.6	N/A	20.1X
ckt4	16.7M	N/A	495	259.4	62	12.5	N/A	20.8X
ckt5	18.8M	N/A	536	314.2	62	14.6	N/A	21.5X
ckt6	19.9M	N/A	540	331.6	59	15.2	N/A	21.8X
ckt7	20.9M	N/A	551	359.5	61	16.2	N/A	22.2X

One other salient feature of the proposed preconditioners (apart from the near-optimal complexity of solving the systems $\mathbf{Mz} = \mathbf{r}$ and the parallelization opportunities) is that there is no need for explicit storage of the preconditioner matrix $\mathbf{M}$, which comes in contrast with most standard preconditioners. As it is easily observed, only the eigenvalues and the values γ_i and δ_i of $\mathbf{M}$ matrices in (4) and (11) respectively are necessary in the execution of algorithms from Fig. 2 and Fig. 3. Thus, only limited storage is required for the preconditioners. A small memory footprint is very important for mapping the algorithm onto architectures with limited available memory space such as GPUs.

V. EXPERIMENTAL EVALUATION

To evaluate the efficiency of the proposed methodology for combined electro-thermal simulation, we compared three methods for solving the linear systems for the power and the thermal grid (steps 1 and 3 in Fig. 1): the PCG method with zero-fill Incomplete Cholesky preconditioner (ICCG), the proposed method of using PCG with the Fast Transform preconditioners (ET-FTCG), and CHOLMOD [5] which is a state-of-the-art direct solver for sparse SPD linear systems. Each method was ported on a GPU platform and the only part that is executed on the CPU is the construction of the power and thermal grid preconditioners for ET-FTCG and ICCG. Subsequently, the CPU is responsible for transferring the appropriate data to the GPU. We have used the CUDA library [1] (version 4.2, along with CUBLAS, CUSPARSE and CUFFT libraries) for mapping the ICCG and the ET-FTCG algorithm on the GPU.

Due to the lack of a set of available benchmarks for electro-thermal analysis, we have created a set of synthetic benchmarks, based on the modeling described in [10] (namely 10% of the wiring resources are used for the power grid), with size ranging from 3.1M to 20.9M-nodes. For the thermal grid, the length Δ_z of the grid rectangle was selected equal to the layer thickness (which can be variable), while the lengths Δ_x and Δ_y were chosen equal to the smallest routing width/pitch within a layer. We executed all experiments on a Linux workstation, comprising an Intel Core i7 processor running at 2.4GHz (6 cores and 24GB main memory) and an NVIDIA Tesla C2075 GPU with 5GB of main memory. Table I presents the results from the evaluation of the aforementioned methods on the set of benchmark circuits. The number of nodes in each circuit ($Nodes$) refers to the total number of nodes in the power grid, while execution time ($Time$) refers to the average time required for solution at each electro-thermal loop iteration, including any overhead for matrix factorization (in CHOLMOD) and preconditioner construction (in iterative methods).

As we can observe, CHOLMOD (the direct solution method) was able to simulate only the smaller benchmark circuit. Due to its excessive memory requirements, analysis of larger benchmarks was infeasible. On the other hand, both ICCG and ET-FTCG owing to the limited memory requirements were able to simulate the complete set of benchmarks. Moreover, they achieved a speed-up of $7X$ and $66.1X$ respectively.

If we restrict our comparison to the iterative methods, we can observe that the proposed method was able to greatly reduce the number of iterations required for convergence. This is a testament to the efficiency of the proposed preconditioning mechanism. Compared with general purpose preconditioning methods such as Incomplete Cholesky factorization, the proposed preconditioners take into account the topology characteristics of the power and the thermal grid. As a result, they are able to approximate them faithfully enough and reduce the required number of iterations. Moreover, owing to their inherent parallelism, the proposed preconditioners can utilize the vast amount of computational resources found in massively parallel architectures, such as GPUs. Thus, their efficacy is increased with the increasing circuit size. ET-FTCG was able to achieve a speed-up ranging between $9.4X$ and $22.2X$ over ICCG for our benchmark circuits. On the contrary, ICCG was not able to fully utilize the GPU resources due to the limited parallelism found in the triangular solution algorithm.

VI. CONCLUSIONS

We have presented a new simulation approach for combined electrical-thermal simulation. Our method combines a state-of-the-art iterative solution method with a preconditioning mechanism that can tackle the electrical-thermal simulation of power delivery networks found in contemporary nano-scale ICs. Owing to its special formulation that allows utilization of Fast Transform solvers, the proposed preconditioning mechanism is able to harness the computational capabilities of massively parallel architectures. Experimental evaluation of the proposed method on a set of benchmark circuits with size ranging from $3.1M$ to $20.9M$-nodes showed that ET-FTCG achieved a speed-up ranging between $9.4X$ and $22.2X$ over a preconditioned iterative method with incomplete factorization preconditioner when GPUs are utilized.

ACKNOWLEDGMENTS

We would like to thank the anonymous reviewers for their comments. The first author is supported by a grant from 'Bodossaki' public benefit foundation.

REFERENCES

[1] *NVIDIA CUDA Programming Guide, CUSPARSE, CUBLAS, and CUFFT Library User Guides*. [Online]. Available: http://developer.nvidia.com/nvidia-gpu-computing-documentation

[2] O. Axelsson and A. Barker, *Finite Element Solution of Boundary Value Problems. Theory and Computation*. Academic Press, 1984.

[3] K. Banerjee and A. Mehrotra, "Global (interconnect) warming," *Circuits and Devices Magazine, IEEE*, vol. 17, no. 5, pp. 16–32, 2001.

[4] T.-H. Chen and C. C.-P. Chen, "Efficient Large-Scale Power Grid Analysis Based on Preconditioned Krylov-Subspace Iterative Methods," in *ACM/IEEE Design Automation Conf.*, 2001.

[5] Y. Chen, T. A. Davis, W. W. Hager, and S. Rajamanickam, "Algorithm 887: CHOLMOD, Supernodal Sparse Cholesky Factorization and Update/Downdate," *ACM Trans. Math. Softw.*, vol. 35, no. 3, pp. 22:1–22:14, 2008.

[6] C. C. Christara, "Quadratic Spline Collocation Methods for Elliptic Partial Differential Equations," *BIT Numerical Mathematics*, vol. 34, no. 1, pp. 33–61, 1994.

[7] Z. Cui-xiang, H. Guo-qiang, and H. Ming-he, "Some New Parallel Fast Fourier Transform Algorithms," in *Int. Conf. on Parallel and Distributed Computing, Applications and Technologies*, 2005.

[8] Z. Feng, Z. Zeng, and P. Li, "Parallel On-Chip Power Distribution Network Analysis on Multi-Core-Multi-GPU Platforms," *IEEE Trans. VLSI Syst.*, vol. 19, no. 10, pp. 1823–1836, 2011.

[9] X.-X. Liu, Z. Liu, S.-D. Tan, and J. Gordon, "Full-chip Thermal Analysis of 3D ICs with Liquid Cooling by GPU-accelerated GMRES Method," in *Quality Electronic Design (ISQED)*, 2012.

[10] S. Nassif, "Power Grid Analysis Benchmarks," in *Asia and South Pacific Design Automation Conf.*, 2008.

[11] J. Shi, Y. Cai, S. X.-D. Tan, J. Fan, and X. Hong, "Pattern-Based Iterative Method for Extreme Large Power/Ground Analysis," *IEEE Trans. Computer-Aided Design*, vol. 26, no. 4, pp. 680–692, 2007.

[12] C. Van Loan, *Computational Frameworks for the Fast Fourier Transform*. SIAM, 1992.

[13] J. Xie and M. Swaminathan, "Electrical-Thermal Co-Simulation of 3D Integrated Systems With Micro-Fluidic Cooling and Joule Heating Effects," *Components, Packaging and Manufacturing Technology, IEEE Transactions on*, vol. 1, no. 2, pp. 234–246, 2011.

[14] Y. Zhang, J. Cohen, and J. D. Owens, "Fast Tridiagonal Solvers on the GPU," in *ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, 2010.

[15] Y. Zhong and M. D. F. Wong, "Thermal-Aware IR Drop Analysis in Large Power Grid," in *ISQED'08*, 2008.