technologies

MDPI

Article

A Preconditioned Iterative Approach for Efficient Full Chip Thermal Analysis on Massively Parallel Platforms †

George Floros [1,*], Konstantis Daloukas [1,2], Nestor Evmorfopoulos [1] and George Stamoulis [1]

[1] Department of Electrical & Computer Engineering, University of Thessaly, 38221 Volos, Greece; kodalouk@e-ce.uth.gr (K.D.); nestevmo@e-ce.uth.gr (N.E.); georges@e-ce.uth.gr (G.S.)
[2] Helic Inc., 2350 Mission College Boulevard, Suite 495, Santa Clara, CA 95054, USA
* Correspondence: gefloros@e-ce.uth.gr; Tel.: +30-2421-074-979
† This paper is an extended version of our paper published in the Proceedings of the 7th International Conference on Modern Circuit and System Technologies on Electronics and Communications (MOCAST 2018), Thessaloniki, Greece, 7–9 May 2018.

Received: 1 November 2018; Accepted: 17 December 2018; Published: 20 December 2018

Abstract: Efficient full-chip thermal simulation is among the most challenging problems facing the EDA industry today, especially for modern 3D integrated circuits, due to the huge linear systems resulting from thermal modeling approaches that require unreasonably long computational times. While the formulation problem, by applying a thermal equivalent circuit, is prevalent and can be easily constructed, the corresponding 3D equations network has an undesirable time-consuming numerical simulation. Direct linear solvers are not capable of handling such huge problems, and iterative methods are the only feasible approach. In this paper, we propose a computationally-efficient iterative method with a parallel preconditioned technique that exploits the resources of massively-parallel architectures such as Graphic Processor Units (GPUs). Experimental results demonstrate that the proposed method achieves a speedup of 2.2× in CPU execution and a 26.93× speedup in GPU execution over the state-of-the-art iterative method.

Keywords: thermal analysis; integrated circuits; electronic design automation

1. Introduction

The evolution of the manufacturing technology of Integrated Circuits (ICs) has continued unabated over the past fifty years, according to the predictions of Moore's law and has led to extremely complex circuits (modern processors contain several billion transistors and are easily the most complex human construction), but also to analogous escalation of the problems related to the analysis and simulation of such circuits. Therefore, thermal analysis is one of the most critical challenges arising from the technological evolution. The continuous effort for smaller sizes, in the sub-45-nm era, and greater performance, as well as the new 3D structures have begun to outpace the ability of heat sinks to dissipate the on-chip power.

In particular, aggravation of thermal effects is an inevitable consequence of the continuous scaling trend. High temperature has a significant impact on chip performance and functionality, leading to slower transistor speed, more leakage power, higher interconnect resistance, and reduced reliability [1]. The problem becomes more pronounced in modern technologies due to the multilayer 3D stacking, and the use of new device technologies, like FinFETsand Silicon on Insulator (SOI), which are more sensitive to the self-heating effect [2]. Furthermore, as heat generation is nonuniform, local hotspots and spatial gradients appear. Stacking multiple layers in a 3D chip promises density and performance enhancement. However, it requires extensive thermal analysis as the power density and temperature

of these architectures can be quite high. For the above reasons, full chip thermal analysis is a vital, but extremely difficult problem due to the size of the systems that need to be solved for multiple time points and remains a key issue for future microprocessors and ICs [3,4]. Due to this fact, IC thermal analysis problems have drawn considerable attention over the past two decades. To deal with these challenges, prior approaches have focused on the formulation of the problem and the fast steady-state and transient thermal simulation in order to compute the temperature across the whole chip.

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 are prohibitively expensive, while the thermal problems are becoming larger, in both execution time and memory requirements. On the other hand, iterative Krylov-subspace methods such as Conjugate Gradients (CG) 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.

Moving beyond conventional direct solvers, our early work in [5] introduced an approach for full chip thermal analysis that is based on the Finite Difference Method (FDM) or the formulation of an RC equivalent electrical network in conjunction with a highly parallel iterative Krylov-subspace Preconditioned Conjugate Gradient (PCG) method, which overcomes the computational demands for the very large systems arising from the thermal modeling. In particular, the contributions of this paper to the problem of thermal analysis are:

- Accelerated solution of thermal grids: The proposed thermal simulator uses FDM with preconditioned CG, which is well-suited, offers faster solution times, and uses less memory than sparse direct solvers.
- Highly parallel preconditioned mechanism: The specialized structures of thermal grids allow the usage of fast transform solvers as a preconditioned mechanism in the CG method, which are highly parallel and can be easily ported to GPUs.
- Fast convergence to the solution: Fast transform solvers can handle different matrix blocks, which offers a good preconditioner approximation. This results in considerably more accurate preconditioners that can approximate thermal grids and make CG converge to the final solution in a few iterations.

Experimental results demonstrate that our method achieves speedups of around 20× on GPU and around 2× on CPU for a 10 M node thermal grid over a state-of-the-art iterative method, like Incomplete Cholesky Preconditioned Conjugate Gradient (ICCG) on a CPU.

The rest of the paper, is organized as follows. Section 2 describes the related work on the thermal simulation problem. Section 3 introduces the thermal model that was used in the present work. Section 4 provides a brief description of the 3D fast transform solver. Section 5 describes the proposed approach, combining the methods presented in the two previous sections. Finally, Section 6 presents the results and a discussion about the advantages of the method, followed by the conclusions in Section 7.

2. Related Work

The growing need to simulate large-scale thermal models in technology nodes below 45 nm has led to some important research in the fast thermal estimation of the IC chips. In this section, we briefly review some of these methods. Most transient thermal analysis methodologies have so far relied on solving the entire system, using different modeling techniques, based mainly on the Finite Element Method (FEM), the Finite Difference Method (FDM), and Green's functions. The research work in [6,7] adopted the FDM method, with a multigrid approach in order to speed up the simulation process, and the FDM method with temporal and spatial adaptation to further accelerate thermal analysis was proposed in [8,9]. Similarly, in [10], the full-chip thermal transient equations were solved in a similar manner using an Alternating Direction Implicit (ADI) method for enhanced computational efficiency. Furthermore, in [11,12], the FDM approach and the RCequivalent were used along with

modeling of the fluids for micro-cooling 3D structures. In [13], FEM was adopted for 2D and 3D geometries along with a multigrid preconditioning method and automatic mesh generation for chip geometries. Finally, Green's functions were used in [14] with discrete cosine transform and its inversion in order to accelerate the numerical computation of the homogeneous and inhomogeneous solution. However, these method are efficient for a limited range of problems, since they have limited potential for parallelism.

Besides the previous conventional approaches, different methods like a Neural Net (NN) approach were used in [15], but since it was based in predictions, it did not always provide an accurate solution to the crucial problem of thermal analysis. Moreover, a Look Up Table (LUT) method based on the power thermal relation, which develops a double-mesh scheme to capture thermal characteristics and store the results in library files, was presented in [16]. However, currently, chips can lead to huge library files due to the highly complex combined heat maps. Furthermore, the reduction of the problem size, through a Model Order Reduction (MOR) process, was proposed in [17,18], which can be useful in addressing the performance of individual devices, but in some cases, it is not enough to address all the reliability issues.

Finally, the authors in [19] provided a parallel iterative Generalized Minimal RESidual (GMRES) method for FDM and micro-cooling problems, but without any special preconditioning approach. Clearly, the concept of a dedicated fully-parallel preconditioned technique has not yet been introduced in the context of transient thermal analysis.

3. On-Chip Thermal Modeling and Analysis

There are three modes of heat transfer: conduction, convection, and radiation. The primary mechanism of heat transfer in solids is by conduction, and the others can be neglected. The starting point for thermal analysis is Fourier's law of heat conduction [20]:

$$\mathbf{q}(\mathbf{r},t) = -k_t \nabla T(\mathbf{r},t) \tag{1}$$

which states that the vector of heat flux density $\mathbf{q}$ (heat flow per unit area and unit time) is proportional to the negative gradient of temperature T at every spacial point $\mathbf{r} = [x,y,z]^T$ and time t, where k_t is the thermal conductivity of the material.

The conservation of energy also states that the divergence of the heat flux $\mathbf{q}$ equals the difference between the power generated by external heat sources and the rate of change of temperature, i.e.,

$$\nabla \cdot \mathbf{q}(\mathbf{r},t) = g(\mathbf{r},t) - \rho c_p \frac{\partial T(\mathbf{r},t)}{\partial t} \tag{2}$$

where $g(\mathbf{r},t)$ is the power density of the heat sources, c_p is the specific heat capacity of the material, and ρ is the density of the material. By combining (1) and (2), we have:

$$-k_t \nabla^2 T(\mathbf{r},t) = g(\mathbf{r},t) - \rho c_p \frac{\partial T(\mathbf{r},t)}{\partial t} \tag{3}$$

which may be rewritten as the following parabolic Partial Differential Equation (PDE):

$$\begin{aligned} \rho c_p \frac{\partial T(\mathbf{r},t)}{\partial t} &= k_t \nabla^2 T(\mathbf{r},t) + g(\mathbf{r},t) \\ = k_t \left(\frac{\partial^2 T(\mathbf{r},t)}{\partial x^2} + \frac{\partial^2 T(\mathbf{r},t)}{\partial y^2} + \frac{\partial^2 T(\mathbf{r},t)}{\partial z^2} \right) &+ g(\mathbf{r},t) \end{aligned} \tag{4}$$

(normally accompanied by appropriate boundary conditions [21]).

A common procedure for the numerical solution of (4) is by discretization along the three spatial coordinates with steps Δx, Δy, and Δz and substitution of the spatial second-order derivatives by finite

difference approximations, leading to the following expression for temperature $T_{i,j,k}$ at each discrete point (i, j, k) in relation to its neighboring points:

$$\begin{aligned} \rho c_p \frac{dT_{i,j,k}}{dt} &= k_t \frac{T_{i+1,j,k} - 2T_{i,j,k} + T_{i-1,j,k}}{\Delta x^2} \\ &+ k_t \frac{T_{i,j+1,k} - 2T_{i,j,k} + T_{i,j-1,k}}{\Delta y^2} \\ &+ k_t \frac{T_{i,j,k+1} - 2T_{i,j,k} + T_{i,j,k-1}}{\Delta z^2} + g_{i,j,k} \end{aligned} \tag{5}$$

or by multiplying by $\Delta x \Delta y \Delta z$:

$$\begin{aligned} &\rho c_p (\Delta x \Delta y \Delta z) \frac{dT_{i,j,k}}{dt} \\ &- k_t \frac{\Delta y \Delta z}{\Delta x} (T_{i+1,j,k} - 2T_{i,j,k} + T_{i-1,j,k}) \\ &- k_t \frac{\Delta x \Delta z}{\Delta y} (T_{i,j+1,k} - 2T_{i,j,k} + T_{i,j-1,k}) \\ &- k_t \frac{\Delta x \Delta y}{\Delta z} (T_{i,j,k+1} - 2T_{i,j,k} + T_{i,j,k-1}) \\ &= g_{i,j,k} (\Delta x \Delta y \Delta z) \end{aligned} \tag{6}$$

There is a well-known analogy between thermal and electrical conduction, where temperature corresponds to voltage and heat flow corresponds to current (see Table 1).

Table 1. Analogy between electrical and thermal circuits.

Electrical Circuit	Thermal Circuit
Voltage	Temperature
Current	Heat Flow
Electrical Conductance	Thermal Conductance
Electrical Resistance	Thermal Resistance
Electrical Capacitance	Thermal Capacitance
Current Source	Heat Source

In light of this analogy, Equation (6) has a direct correspondence to an electrical circuit where there is a node at every discrete point or cell in the thermal grid (see Figure 1). Every circuit node is connected to spatially-neighboring nodes via conductances in the directions x, y, z with values:

$$G_x \equiv \frac{k_t \Delta y \Delta z}{\Delta x}, G_y \equiv \frac{k_t \Delta x \Delta z}{\Delta y}, G_z \equiv \frac{k_t \Delta x \Delta y}{\Delta z} \tag{7}$$

and there is a capacitance to ground at every node or thermal cell with value:

$$C \equiv \rho c_p (\Delta x \Delta y \Delta z) \tag{8}$$

The heat sources constitute input excitations and are modeled in the equivalent circuit as the current sources with values:

$$I_{i,j,k} \equiv g_{i,j,k} (\Delta x \Delta y \Delta z) \tag{9}$$

The above current sources are connected at the specific points (i, j, k) or circuit nodes where there is heat flow (i.e., power dissipation from the underlying chip logic blocks).

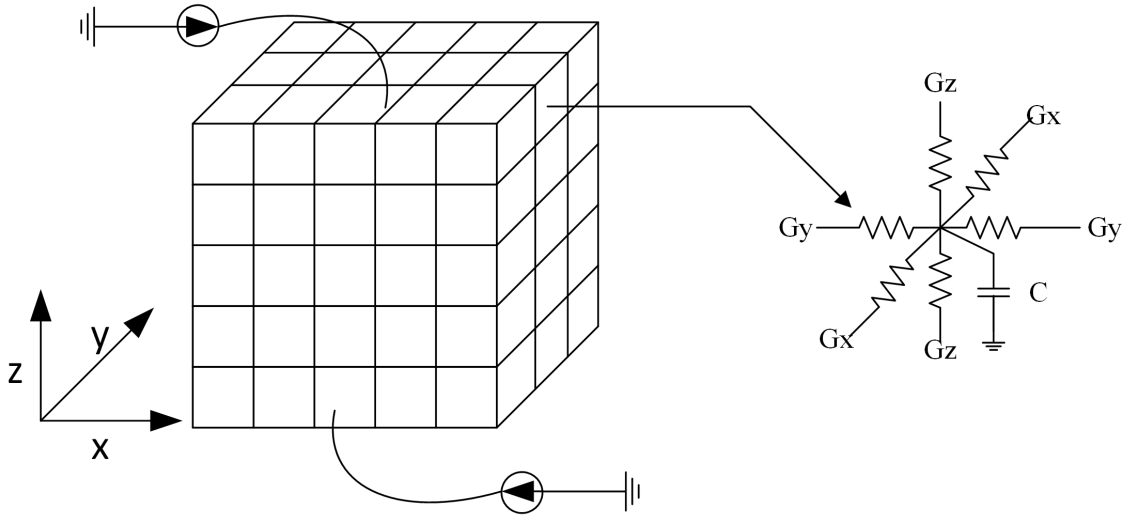


Figure 1. Spatial discretization of a chip for thermal analysis and the formulation of the electrical equivalent problem.

The resulting electrical equivalent circuit is described in the time domain, using the Modified Nodal Analysis (MNA) framework, by a system of Ordinary Differential Equations (ODE):

$$\mathbf{G}\mathbf{x}(t) + \mathbf{C}\frac{d\mathbf{x}(t)}{dt} = \mathbf{u}(t) \tag{10}$$

where $\mathbf{G} \in \mathcal{R}^{n\times n}$ is a symmetric and positive definite matrix of the conductances (7), $\mathbf{C} \in \mathcal{R}^{n\times n}$ is a diagonal matrix of cell capacitances (8), $\mathbf{x} \in \mathcal{R}^n$ is the vector of unknown temperatures $T_{i,j,k}$ at all discretization points (constituting internal states of the system), and $\mathbf{u} \in \mathcal{R}^p$ is the vector of input excitations from the current sources $I_{i,j,k}$ of (9).

For transient simulation, we can discretize the time interval into time instants t_k, $k = 1, 2, \ldots$ and use the backward-Euler numerical integration method for the calculation of temperature in each discrete time instant t_k:

$$(\mathbf{G} + \frac{\mathbf{C}}{h_k})\mathbf{x}(t_k) = \frac{\mathbf{C}}{h_k}\mathbf{x}(t_{k-1}) + \mathbf{u}(t_k) \tag{11}$$

where $h_k = t_k - t_{k-1}$, $k = 1, 2, \ldots$ is the time step of time t_k (which may in general vary during transient analysis). The above equation involves the solution of a very large sparse linear system in each time instant t_k.

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. 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. In this work, we employ iterative methods for the solution of large-scale networks arising from the 3D discretization of the chip for thermal analysis. The system matrices arising from the modeling of the thermal grid can also be shown to be Symmetric and Positive Definite (SPD), which allows the use of the efficient method of the Conjugate Gradient (CG) for the solution of the corresponding linear systems. The CG method is well known [22], and its implementation is shown in Algorithm 1.

Algorithm 1 Preconditioned conjugate gradient.

$\mathbf{x} =$ initial guess $\mathbf{x}^{(0)}$
$\mathbf{r} = \mathbf{b} - \mathbf{A}\mathbf{x}$
$iter = 0$
repeat
  $iter = iter + 1$
  Solve $\mathbf{M}\mathbf{z} = \mathbf{r}$ (**Preconditioner-Solve Step**)
  $\rho = \mathbf{r} \cdot \mathbf{z}$
  if $iter == 1$ **then**
    $\mathbf{p} = \mathbf{z}$
  else
    $\beta = \rho / \rho 1$
    $\mathbf{p} = \mathbf{z} + \beta \mathbf{p}$
  end if
  $\rho 1 = \rho$
  $\mathbf{q} = \mathbf{A}\mathbf{p}$
  $\alpha = \rho / (\mathbf{p} \cdot \mathbf{q})$
  $\mathbf{x} = \mathbf{x} + \alpha \mathbf{p}$
  $\mathbf{r} = \mathbf{r} - \alpha \mathbf{q}$
until $\frac{||\mathbf{b}-\mathbf{A}\mathbf{x}||}{||\mathbf{b}||} < tol$

Regarding the convergence rate of CG, it can be shown [23] that the required number of iterations (for a given initial guess and convergence tolerance) is bounded in terms of the spectral condition number $k_2(\mathbf{A}) = ||\mathbf{A}||_2 ||\mathbf{A}^{-1}||_2 \geq 1$; specifically, it is $O(\sqrt{k_2(\mathbf{A})})$, which for SPD matrices becomes $k_2(\mathbf{A}) = \frac{\lambda_{max}(\mathbf{A})}{\lambda_{min}(\mathbf{A})}$ where $\lambda_{max}(\mathbf{A})$, $\lambda_{min}(\mathbf{A})$ are the maximum and minimum eigenvalues of $\mathbf{A}$, respectively. This means that convergence of CG is fast when $k_2(\mathbf{A}) \approx 1$ and slow when $k_2(\mathbf{A}) >> 1$.

To improve the convergence speed, it is necessary to apply a preconditioning mechanism, which transforms the initial linear system into an equivalent one with a more favorable spectral condition number. The so-called preconditioner is a matrix $\mathbf{M}$ that approximates $\mathbf{A}$ in some way, such that the transformed system $\mathbf{M}^{-1}\mathbf{A}\mathbf{x} = \mathbf{M}^{-1}\mathbf{b}$ (which obviously has the same solution as the initial $\mathbf{A}\mathbf{x} = \mathbf{b}$) exhibits condition number $k_2(\mathbf{M}^{-1}\mathbf{A}) = k_2(\mathbf{I}) = 1$. In practice, it is not necessary to invert the preconditioner $\mathbf{M}$ and apply it directly at the system $\mathbf{A}\mathbf{x} = \mathbf{b}$. It can be shown that the same thing can be accomplished by introducing an extra computational step within the iterative method, which entails solving a system $\mathbf{M}\mathbf{z} = \mathbf{r}$ with known Right-Hand Side (RHS) vector $\mathbf{r}$ and unknown vector $\mathbf{z}$ in every iteration [23].

From the above, it follows that a good preconditioner $\mathbf{M}$ must satisfy two key properties:

- The fast convergence rate of the preconditioned.
- A linear system involving $\mathbf{M}$ is solved much more efficiently than the original system that involves $\mathbf{A}$.

where "more efficiently" can mean with less asymptotic complexity—ideally, an optimal or near-optimal complexity of $O(N)$ or $O(NlogN)$—and/or significantly more parallelism in the solution procedure. If the preconditioner is faithful enough to reduce the iterations substantially, then the whole burden of the algorithm is transferred to the preconditioner-solve step $\mathbf{M}\mathbf{z} = \mathbf{r}$. The next section will describe the proposed form of the preconditioner matrices for 3D thermal networks, as well as the solution of the corresponding linear systems via two series of fast transforms and inverse fast transforms.

4. Fast Transform Preconditioners for 3D Thermal Networks

Recent implementations of fast transform solvers have shown great potential for the solution of block-tridiagonal systems with a special structure [24,25]. This section describes such an algorithm for

the solution of an appropriate preconditioner system $\mathbf{Mz} = \mathbf{r}$ by the use of a fast transform solver in a near-optimal number of operations.

Let $\mathbf{M}$ be an $N \times N$ block-tridiagonal matrix with l diagonal blocks of size $mn \times mn$ each (overall $N = lmn$), where l is very small (typically 5–8 depending on the material layers (metal and insulator) of the chip), with the following form:

$$\mathbf{M} = \begin{bmatrix} \mathbf{M}_1 & -\delta_1\mathbf{I}_{mn} & & & \\ -\delta_1\mathbf{I}_{mn} & \mathbf{M}_2 & -\delta_2\mathbf{I}_{mn} & & \\ & \cdot & \cdot & \cdot & \\ & & -\delta_{l-2}\mathbf{I}_{mn} & \mathbf{M}_{l-1} & -\delta_{l-1}\mathbf{I}_{mn} \\ & & & -\delta_{l-1}\mathbf{I}_{mn} & \mathbf{M}_l \end{bmatrix} \tag{12}$$

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

$$\mathbf{M}_i = \begin{bmatrix} \mathbf{T}_i + \gamma_i\mathbf{I}_n & -\gamma_i\mathbf{I}_n & & & \\ -\gamma_i\mathbf{I}_n & \mathbf{T}_i + 2\gamma_i\mathbf{I}_n & -\gamma_i\mathbf{I}_n & & \\ & \cdot & \cdot & \cdot & \\ & & -\gamma_i\mathbf{I}_n & \mathbf{T}_i + \gamma_i\mathbf{I}_n & -\gamma_i\mathbf{I}_n \\ & & & -\gamma_i\mathbf{I}_n & \mathbf{T}_i + \gamma_i\mathbf{I}_n \end{bmatrix} \tag{13}$$

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 with the following form:

$$\mathbf{T}_i = \begin{bmatrix} \alpha_i + \beta_i & -\alpha_i & & & \\ -\alpha_i & 2\alpha_i + \beta_i & -\alpha_i & & \\ & \cdot & \cdot & \cdot & \\ & & -\alpha_i & 2\alpha_i + \beta_i & -\alpha_i \\ & & & -\alpha_i & \alpha_i + \beta_i \end{bmatrix} =$$

$$\alpha_i \begin{bmatrix} 1 & -1 & & & \\ -1 & 2 & -1 & & \\ & \cdot & \cdot & \cdot & \\ & & -1 & 2 & -1 \\ & & & -1 & 1 \end{bmatrix} + \beta_i\mathbf{I} \tag{14}$$

This class of tridiagonal matrices has a beforehand known eigen-decomposition. Specifically, it can be shown [26] that each $\mathbf{T}_i$ has n distinct eigenvalues $\lambda_{i,j}, j = 1, \ldots, n$, which are given by:

$$\lambda_{i,j} = \beta_i + 4\alpha_i sin^2(\frac{(j-1)\pi}{2n}) = \beta_i + \alpha_i(2cos(\frac{(j-1)\pi}{n}) - 2) \tag{15}$$

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

$$q_{j,k} = \begin{cases} \sqrt{\frac{1}{n}}\cos\frac{(2k-1)(j-1)\pi}{2n}, & j = 1, \quad k = 1, \ldots, n \\ \sqrt{\frac{2}{n}}\cos\frac{(2k-1)(j-1)\pi}{2n}, & j = 2, \ldots, n, \quad k = 1, \ldots, n \end{cases} \tag{16}$$

Note that, the eigenvectors do not depend on the values 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 the

diagonalization of the matrix $\mathbf{T}_i$ and considering that $\mathbf{Q}_n^T\mathbf{Q}_n = \mathbf{I}$, the system $\mathbf{Mz} = \mathbf{r}$ is equivalent to the following system:

$$\begin{aligned}\begin{bmatrix} \mathbf{Q}_n^T & & \\ & \ddots & \\ & & \mathbf{Q}_n^T \end{bmatrix} \mathbf{M} \begin{bmatrix} \mathbf{Q}_n & & \\ & \ddots & \\ & & \mathbf{Q}_n \end{bmatrix} \begin{bmatrix} \mathbf{Q}_n^T & & \\ & \ddots & \\ & & \mathbf{Q}_n^T \end{bmatrix} \mathbf{z} \\ = \begin{bmatrix} \mathbf{Q}_n^T & & \\ & \ddots & \\ & & \mathbf{Q}_n^T \end{bmatrix} \mathbf{r} \Leftrightarrow\end{aligned} \tag{17}$$

$$\begin{bmatrix} \tilde{\mathbf{M}}_1 & -\delta_1\mathbf{I}_{mn} & & & & \\ -\delta_1\mathbf{I}_{mn} & \tilde{\mathbf{M}}_2 & -\delta_2\mathbf{I}_{mn} & & & \\ & \cdot & \cdot & \cdot & & \\ & & -\delta_{l-2}\mathbf{I}_{mn} & \tilde{\mathbf{M}}_{l-1} & -\delta_{l-1}\mathbf{I}_{mn} & \\ & & & -\delta_{l-1}\mathbf{I}_{mn} & \tilde{\mathbf{M}}_l & \end{bmatrix} \tilde{\mathbf{z}} = \tilde{\mathbf{r}} \tag{18}$$

where:

$$\tilde{\mathbf{M}}_\mathbf{i} = \begin{bmatrix} \mathbf{\Lambda}_i^{(1)} & -\gamma_i\mathbf{I} & & & \\ -\gamma_i\mathbf{I} & \mathbf{\Lambda}_i^{(2)} & -\gamma_i\mathbf{I} & & \\ & \cdot & \cdot & \cdot & \\ & & -\gamma_i\mathbf{I} & \mathbf{\Lambda}_i^{(2)} & -\gamma_i\mathbf{I} \\ & & & -\gamma_i\mathbf{I} & \mathbf{\Lambda}_i^{(1)} \end{bmatrix} \tag{19}$$

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

and $\mathbf{\Lambda}_i^{(1)} = diag(\lambda_{i,1}^{(1)}, \ldots, \lambda_{i,n}^{(1)})$ and $\mathbf{\Lambda}_i^{(2)} = diag(\lambda_{i,1}^{(2)}, \ldots, \lambda_{i,n}^{(2)})$ 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:

$$\begin{aligned}\lambda_{i,j}^{(1)} &= \gamma_i + \beta_i + \alpha_i(2cos(\frac{(j-1)\pi}{n}) - 2) \quad j = 1, \ldots, n, \\ \lambda_{i,j}^{(2)} &= 2\gamma_i + \beta_i + \alpha_i(2cos(\frac{(j-1)\pi}{n}) - 2) \quad j = 1, \ldots, n,\end{aligned} \tag{20}$$

If the $N \times 1$ vectors $\mathbf{r}$, $\mathbf{z}$, $\tilde{\mathbf{r}}$, $\tilde{\mathbf{z}}$ are also partitioned into m 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 [27] 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 the Fast Fourier Transform (FFT) can be employed for each of the lm independent DCT-II/IDCT-II transforms [27], giving a total near-optimal operation count of $\mathcal{O}(mn \log n) = \mathcal{O}(N \log n)$.

If now, $\mathbf{P}$ is a permutation matrix of size $mn \times mn$ 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 $\mathbf{P}_1$,

$\mathbf{P}_1^T$ are 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 at (18) is transformed into:

$$\mathbf{P}_1 \begin{bmatrix} \tilde{\mathbf{M}}_1 & -\delta_1\mathbf{I}_{mn} & & & & \\ -\delta_1\mathbf{I}_{mn} & \tilde{\mathbf{M}}_2 & -\delta_2\mathbf{I}_{mn} & & & \\ & \cdot & \cdot & \cdot & & \\ & & -\delta_{l-2}\mathbf{I}_{mn} & \tilde{\mathbf{M}}_{l-1} & -\delta_{l-1}\mathbf{I}_{mn} & \\ & & & -\delta_{l-1}\mathbf{I}_{mn} & \tilde{\mathbf{M}}_l & \end{bmatrix} \mathbf{P}_1^T\mathbf{P}_1\tilde{\mathbf{z}} = \mathbf{P}_1\tilde{\mathbf{r}} \Leftrightarrow$$

$$\begin{bmatrix} \mathbf{D}_1 & -\delta_1\mathbf{I}_{mn} & & & & \\ -\delta_1\mathbf{I}_{mn} & \mathbf{D}_2 & -\delta_2\mathbf{I}_{mn} & & & \\ & \cdot & \cdot & \cdot & & \\ & & -\delta_{l-2}\mathbf{I}_{mn} & \mathbf{D}_{l-1} & -\delta_{l-1}\mathbf{I}_{mn} & \\ & & & -\delta_{l-1}\mathbf{I}_{mn} & \mathbf{D}_l & \end{bmatrix} \tilde{\mathbf{z}}^{\mathbf{P}_1} = \tilde{\mathbf{r}}^{\mathbf{P}_1} \quad (21)$$

where $\mathbf{D}_1 = diag(\tilde{\mathbf{T}}_{i,1}, \ldots, \tilde{\mathbf{T}}_{i,n})$, $i = 1, \ldots, l$, with $\tilde{\mathbf{T}}_{i,j}$, $j = 1, \ldots, n$ being $m \times m$ tridiagonal matrices of the form:

$$\tilde{\mathbf{T}}_{i,j} = \begin{bmatrix} \lambda_{i,j}^{(1)} & -\gamma_i & & & \\ -\gamma_i & \lambda_{i,j}^{(2)} & -\gamma_i & & \\ & \cdot & \cdot & \cdot & \\ & & -\gamma_i & \lambda_{i,j}^{(2)} & -\gamma_i \\ & & & -\gamma_i & \lambda_{i,j}^{(1)} \end{bmatrix} =$$

$$\gamma_i \begin{bmatrix} 1 & -1 & & & \\ -1 & 2 & -1 & & \\ & \cdot & \cdot & \cdot & \\ & & -1 & 2 & -1 \\ & & & -1 & 1 \end{bmatrix} + (\beta_i + \alpha_i(2cos(\frac{(j-1)\pi}{n}) - 2))\mathbf{I}_m \quad (22)$$

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:

$$\tilde{\lambda}_{i,j,k} = \gamma_i(2cos(\frac{(k-1)\pi}{n}) - 2) + \beta_i + \alpha_i(2cos(\frac{(j-1)\pi}{n}) - 2), \quad k = 1, \ldots, m \quad (23)$$

and $\mathbf{Q}_m$ is the common matrix of eigenvectors for all $\tilde{\mathbf{T}}_{i,j}$, then by similar reasoning as in (17), the system (21) is equivalent to:

$$\begin{bmatrix} \tilde{\mathbf{D}}_1 & -\delta_1\mathbf{I}_{mn} & & & & \\ -\delta_1\mathbf{I}_{mn} & \tilde{\mathbf{D}}_2 & -\delta_2\mathbf{I}_{mn} & & & \\ & \cdot & \cdot & \cdot & & \\ & & -\delta_{l-2}\mathbf{I}_{mn} & \tilde{\mathbf{D}}_{l-1} & -\delta_{l-1}\mathbf{I}_{mn} & \\ & & & -\delta_{l-1}\mathbf{I}_{mn} & \tilde{\mathbf{D}}_l & \end{bmatrix} \tilde{\tilde{\mathbf{z}}} = \tilde{\tilde{\mathbf{r}}} \quad (24)$$

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

$$\tilde{\tilde{\mathbf{z}}} = \begin{bmatrix} \mathbf{Q}_m^T & & \\ & \ddots & \\ & & \mathbf{Q}_m^T \end{bmatrix} \tilde{\mathbf{z}}_1^{\mathbf{P}}, \quad \tilde{\tilde{\mathbf{r}}} = \begin{bmatrix} \mathbf{Q}_m^T & & \\ & \ddots & \\ & & \mathbf{Q}_m^T \end{bmatrix} \tilde{\mathbf{r}}_1^{\mathbf{P}}$$

In a similar way as previously, the $N \times 1$ vectors $\tilde{\mathbf{z}}^{\mathbf{P}_1}$, $\tilde{\mathbf{r}}^{\mathbf{P}_1}$, $\tilde{\tilde{\mathbf{z}}}$, and $\tilde{\tilde{\mathbf{r}}}$ can be partitioned into ln sub-vectors of size $m \times 1$ each, and the DCT-II and IDCT-II are performed accordingly, giving a total near-optimal operation count of $\mathcal{O}(lmn \log n) = \mathcal{O}(N \log n)$.

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 (24) is equivalent to:

$$\tilde{\tilde{\mathbf{M}}}\tilde{\tilde{\mathbf{z}}}^{\mathbf{P}_2} = \tilde{\tilde{\mathbf{r}}}^{\mathbf{P}_2} \tag{25}$$

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

$$\tilde{\tilde{\mathbf{T}}}_{j,k} = \begin{bmatrix} \tilde{\lambda}_{1,i,k} & -\delta_1 & & & \\ -\delta_1 & \tilde{\lambda}_{2,i,k} & -\delta_2 & & \\ & \cdot & \cdot & \cdot & \\ & & -\delta_{l-1} & \tilde{\lambda}_{l-1,i,k} & -\delta_l \\ & & & -\delta_l & \tilde{\lambda}_{l,i,k} \end{bmatrix} \tag{26}$$

and $\tilde{\tilde{\mathbf{z}}}^{\mathbf{P}_2} = \mathbf{P}_2\tilde{\tilde{\mathbf{z}}}, \tilde{\tilde{\mathbf{r}}}^{\mathbf{P}_2} = \mathbf{P}_2\tilde{\tilde{\mathbf{r}}}$.

Taking into account the above equations and by applying permutation matrices in order to reorder the elements of the $\mathbf{Mz} = \mathbf{r}$ system, a fast solution for the preconditioner solve step can be obtained as shown in Algorithm 2.

Algorithm 2 Preconditioner solution for the thermal grid.

1: Partition 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 subvector 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 $\tilde{\mathbf{r}}_\mathbf{i}^{\mathbf{P}_1}$ of size m, and perform DCT-II transform ($\mathbf{Q}_m^T\tilde{\mathbf{r}}_\mathbf{i}^{\mathbf{P}_1}$) 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 $\tilde{\tilde{\mathbf{r}}}^{\mathbf{P}_2}$.
5: Calculate elements of matrices $\tilde{\mathbf{T}}_{n,m}$, and solve the mn tridiagonal systems, in order to obtain vector $\tilde{\tilde{\mathbf{z}}}^{\mathbf{P}_2}$.
6: Apply inverse permutation $\mathbf{P}_2^T$ on vector $\tilde{\tilde{\mathbf{z}}}^{\mathbf{P}_2}$ 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 $\mathbf{P}^T$ 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 ($\mathbf{Q}_n\mathbf{z}_i$) on each sub-vector to obtain final solution vector $\mathbf{z}$.

5. Methodology for Full Chip Thermal Analysis

This section applies the theoretical background that was analyzed before for the computation of the temperatures across the chip. The complete methodology consists of the following steps:

- 3D discretization of the chip: The spatial steps Δx, Δy in the x- and y-direction are user defined, but the step Δz along the z-direction is typically chosen to coincide with the interface between successive layers (metal and insulator). The discretization procedure naturally covers multiple layers in the z-direction and can be easily extended to model heterogeneous structures that can be found in modern chips (e.g., heat sinks).
- Construction of equivalent electrical circuit: The RC elements of the electrical equivalent are calculated by (7) and (8).
- Estimation of the power consumption profile of chip logic blocks. This determines the location and the time behavior of heat sources and the value of current sources (9) that constitute the vector $\mathbf{u}(t)$ in (10).
- Formulation of equivalent circuit description: Using modified nodal analysis, the equivalent circuit is described by the ODE system (10).
- Construction of the preconditioner matrix: Based on the algorithm described in the previous section, the preconditioner matrix is constructed based on [24], and the preconditioner-solve step is performed with the fast transform solver. More specifically, the thermal grid is equivalent to a highly regular resistive network, as depicted in Figure 2, 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 connecting nodes in adjacent layers (z axis) with their average value between the two layers.
- Compute either the DC or transient solution: The solution is obtained with the iterative PCG method in both cases. Note that in the case of the transient solution, the backward-Euler numerical integration method as in (11) is employed for the calculation of temperature in each discrete time instant. The convergence of the method is accelerated with the highly parallel fast transform solver that is used in preconditioner-solve step in Algorithm 1.

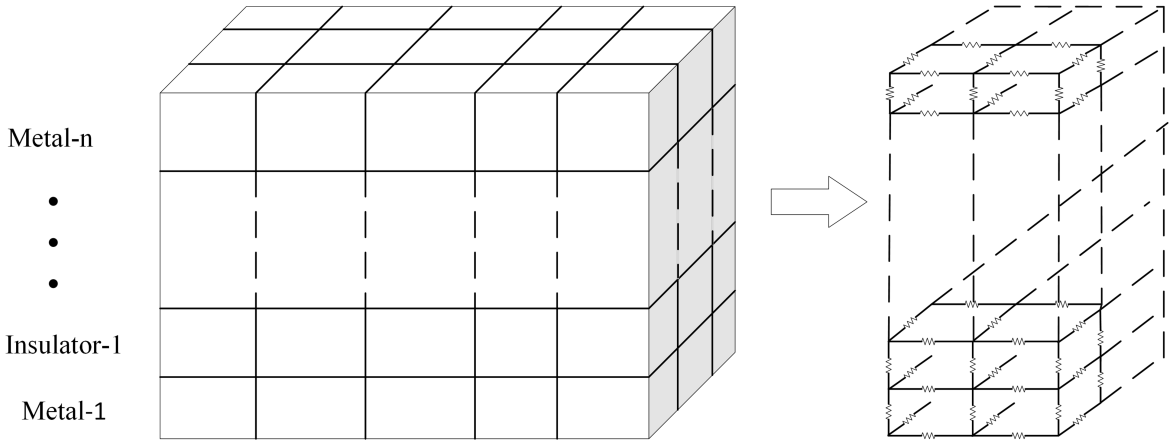


Figure 2. Example of a 3D thermal grid that is used for preconditioning.

The proposed methodology offers significant advantages over the established thermal simulation methods. Firstly, iterative methods can handle large-scale problems in contrast to direct methods that do not scale well with the matrix dimension and can only be applied to a narrow range of problems. Furthermore, the fast preconditioned solution step exhibits near-optimal computational complexity, low memory requirements, and great potential for parallelism, which can harness the computational power of parallel architectures, such as multicore processors or GPUs, thus further reducing the amount of time required for simulation.

6. Experimental Results

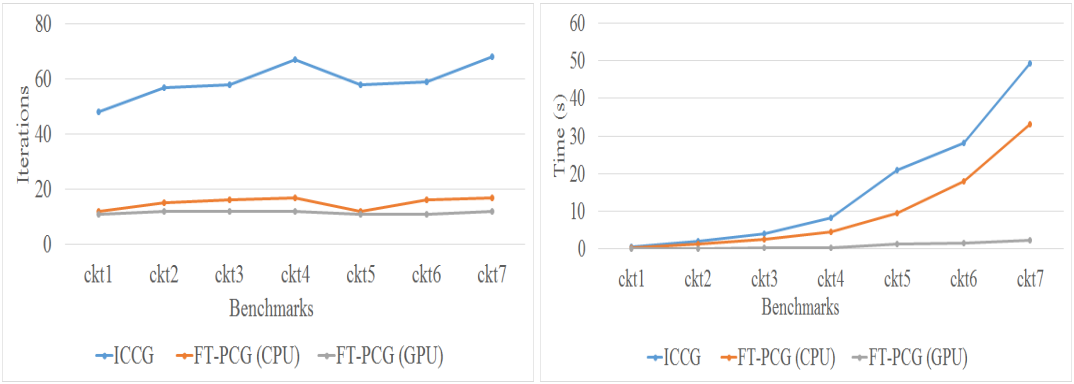
Due to the lack of availability of benchmarks for full chip thermal analysis and in order to evaluate the efficiency of the proposed methodology for thermal simulation, we have created a set of artificial benchmark circuits that represent simplified microprocessor designs (like MIPS and LEON) with a random control logic and datapath, based on the theory described in Section 3. The technology node that was used for this work was 32 nm. Similarly, one can form the linear set of equations for different technology-specific parameters of Equations (7) and (8). Despite the fact that the benchmarks were artificially created, the problems that would arise from a real design would be represented by a similar set of linear equations.

In Table 2 discretization points are the number of discretizations in each axis, while layers are the number of layers that each benchmark has. Moreover, the matrix dimensions can be calculated by multiplying the square of the discretization points by the number of layers. All experiments were executed on a Linux workstation, comprised of an Intel Core i7 processor running at 2.4 GHz (six cores and 24 GB main memory) and an NVIDIA Tesla C2075 GPU with 6 GB of main memory. We have used the CUDA library [28] (Version 5.5, along with the CUBLAS, CUSPARSE, and CUFFT libraries) for mapping the proposed Fast-Transform PCG (FT-PCG) algorithm on the GPU. The ICCG was executed on the CPU since it would not be beneficial to port it on the GPU due many irregular memory transfers. The convergence tolerance (tol) for iterative solvers was set to 10^{-6}, which is typically sufficient to yield perfect accuracy, and convergence was achieved in all cases. Table 2 presents the results from the evaluation of the aforementioned methods on the set of benchmark circuits. The number *Iter.* is the number of iterations that the algorithm needed to converge, while *Time (s)* refers to the time in seconds that were needed to compute the final solution. Both the number of iterations and time represent either the DC solution of the system or the average iterations and time of a time instant in transient analysis.

Table 2. Runtime results for the three solvers. Bench. is the name of the benchmark, Discr. Point. is the number of points that correspond to the x and y axis, Layers is the number of layers of each chip and corresponds to the z axis, Time (s) denotes the average time required for the solution at each iteration, Iter. is the average number of iterations required for the convergence of each iterative method, while Speedup denotes the speedup of our method over the ICCG.

Bench.	Discr. Points	Layers	ICCG		FT-PCG (CPU)			FT-PCG (GPU)		
			Iter.	Time (s)	Iter.	Time (s)	Speedup	Iter.	Time (s)	Speedup
ckt1	175	5	48	0.48	12	0.31	1.54×	11	0.03	16×
ckt2	320	5	57	1.98	15	1.23	1.6×	12	0.08	24.75×
ckt3	410	6	58	4.20	16	2.64	1.59×	12	0.23	18.26×
ckt4	500	7	67	8.35	17	4.51	1.85×	12	0.31	26.93×
ckt5	845	7	58	21.07	12	9.48	2.22×	11	1.34	15.72×
ckt6	946	7	59	28.27	16	17.94	1.57×	11	1.60	17.65×
ckt7	1118	8	68	49.35	17	33.07	1.49×	12	2.39	20.64×

Comparing the iterative methods, it can be observed that the proposed method was able to reduce the number of iterations required for convergence greatly, as shown in Figure 3a. Compared with general purpose preconditioning methods such as ICCG, the proposed preconditioners take into account the topology characteristics of the thermal grid. As a result, they are able to approximate it 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 efficiency is increased with the increasing circuit size, by greatly reducing the runtime for each time-step, as depicted in Figure 3b. FT-PCG was able to achieve a speed-up ranging between 1.5× and 2.2× in CPU execution and 16× and 26.93× in GPU execution over ICCG.



(**a**) (**b**)

Figure 3. (**a**) Average number of iterations, (**b**) Average runtimes for each time-step in each benchmark.

7. Conclusions

In this paper, we presented a fast thermal simulation method based on the RC equivalent, which uses fast transform solvers and preconditioned Krylov-subspace iterative solvers. The preconditioned iterative solvers offer linear scaling in simulation time as the thermal grid is increased. Experimental evaluation of the proposed method on a set of thermal benchmarks with the size ranging from 0.15 M–10 M nodes showed that the proposed methodology achieved a speedup ranging between 15.72× and 26.93× over a preconditioned iterative method with an incomplete Cholesky factorization preconditioner when GPUs are utilized.

Author Contributions: Conceptualization, G.F. and N.E.; data curation, G.F.; funding acquisition, N.E. and G.S.; investigation, G.F.; methodology, G.F., K.D. and N.E.; project administration, N.E. and G.S.; software, K.D.; supervision, N.E. and G.S.; validation, G.F. and K.D.; writing, original draft, G.F.; writing, review and editing, G.F., K.D., N.E., and G.S.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

EDA	Electronic Design Automation
GPU	Graphic Processor Unit
IC	Integrated Circuit
SOI	Silicon on Insulator
FDM	Finite Difference Method
PCG	Preconditioned Conjugate Gradient
ICCG	Incomplete Cholesky Conjugate Gradient
FEM	Finite Element Method
ADI	Alternating Direction Implicit
NN	Neural Net
LUT	Look Up Table
MOR	Model Order Reduction
GMRES	Generalized Minimal RESidual
PDE	Partial Differential Equation
LHS	Left-Hand Side
MNA	Modified Nodal Analysis
ODE	Ordinary Differential Equations
SPD	Symmetric and Positive Definite
CG	Conjugate Gradient

RHS	Right-Hand Side
DCT-II	Discrete Cosine Transform of Type-II
IDCT-II	Inverse Discrete Cosine Transform of Type-II
FFT	Fast Fourier Transform
FT-PCG	Fast Transform Preconditioned Conjugate Gradient

References

1. Waldrop, M M. The Chips Are Down for Moore's Law. *Nat. News* **2016**, *530*, 144–147. [CrossRef] [PubMed]
2. Xu, C.; Kolluri, S.K.; Endo, K.; Banerjee, K. Analytical Thermal Model for Self-Heating in Advanced FinFET Devices With Implications for Design and Reliability. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* **2013**, *32*, 1045–1058.
3. SIA. *International Technology Roadmap for Semiconductors (ITRS) 2015 Edition-ERD*; SIA: Washington, DC, USA, 2015.
4. Pedram, M.; Nazarian, S. Thermal modeling, analysis, and management in VLSI circuits: Principles and methods. *Proc. IEEE* **2006**, *94*, 1487–1501. [CrossRef]
5. Floros, G.; Daloukas, K.; Evmorfopoulos, N.; Stamoulis, G. A parallel iterative approach for efficient full chip thermal analysis. In Proceedings of the 7th International Conference on Modern Circuits and Systems Technologies (MOCAST), Thessaloniki, Greece, 7–9 May 2018; pp. 1–4.
6. Li, P.; Pileggi, L.T.; Asheghi, M.; Chandra, R. Efficient full-chip thermal modeling and analysis. In Proceedings of the IEEE/ACM International Conference on Computer Aided Design, San Jose, CA, USA, 7–11 November 2004; pp. 319–326.
7. Li, P.; Pileggi, L.T.; Asheghi, M.; Chandra, R. IC thermal simulation and modeling via efficient multigrid-based approaches. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* **2006**, *25*, 1763–1776.
8. Yang, Y.; Zhu, C.; Gu, Z.; Shang, L.; Dick, R.P. Adaptive multi-domain thermal modeling and analysis for integrated circuit synthesis and design. In Proceedings of the IEEE/ACM International Conference on Computer Aided Design, San Jose, CA, USA, 5–9 November 2006; pp. 575–582.
9. Yang, Y.; Gu, Z.; Zhu, C.; Dick, R.P.; Shang, L. ISAC: Integrated Space-and-Time-Adaptive Chip-Package Thermal Analysis. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* **2007**, *26*, 86–99. [CrossRef]
10. Wang, T.Y.; Chen, C.C.P. 3-D Thermal-ADI: a linear-time chip level transient thermal simulator. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* **2002**, *21*, 1434–1445. [CrossRef]
11. Sridhar, A.; Vincenzi, A.; Ruggiero, M.; Brunschwiler, T.; Atienza, D. 3D-ICE: Fast compact transient thermal modeling for 3D ICs with inter-tier liquid cooling. In Proceedings of the IEEE/ACM International Conference on Computer-Aided Design, San Jose, CA, USA, 7–11 November 2010; pp. 463–470.
12. Sridhar, A.; Vincenzi, A.; Atienza, D.; Brunschwiler, T. 3D-ICE: A Compact Thermal Model for Early-Stage Design of Liquid-Cooled ICs. *IEEE Trans. Comput.* **2014**, *63*, 2576–2589 [CrossRef]
13. Ladenheim, S.; Chen, Y.C.; Mihajlovic, M.; Pavlidis, V. IC thermal analyzer for versatile 3-D structures using multigrid preconditioned Krylov methods. In Proceedings of the IEEE/ACM International Conference on Computer-Aided Design, Austin, TX, USA, 7–10 November 2016; pp. 1–8.
14. Zhan, Y.; Sapatnekar, S.S. High-Efficiency Green Function-Based Thermal Simulation Algorithms. *IIEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* **2007**, *26*, 1661–1675. [CrossRef]
15. Vincenzi, A.; Sridhar, A.; Ruggiero, M.; Atienza, D. Fast thermal simulation of 2D/3D integrated circuits exploiting neural networks and GPUs. In Proceedings of the IEEE/ACM International Symposium on Low Power Electronics and Design, Fukuoka, Japan, 1–3 August 2011; pp. 151–156.
16. Lee, Y.M.; Pan, C.W.; Huang, P.Y.; Yang, C.P. LUTSim: A Look-Up Table-Based Thermal Simulator for 3-D ICs. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* **2015**, *34*, 1250–1263
17. Wang, T.Y.; Chen, C.C.P. SPICE-compatible thermal simulation with lumped circuit modeling for thermal reliability analysis based on modeling order reduction. In Proceedings of the International Symposium on Signals, Circuits and Systems, San Jose, CA, USA, 22–24 March 2004; pp. 357–362.
18. Floros, G.; Evmorfopoulos, N.; Stamoulis, G. Efficient Hotspot Thermal Simulation Via Low-Rank Model Order Reduction. In Proceedings of the 15th International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD), Prague, Czech Republic, 2–5 July 2018; pp. 205–208.

19. Liu, X.X.; Zhai, K.; Liu, Z.; He, K.; Tan, S.X.D.; Yu, W. Parallel Thermal Analysis of 3-D Integrated Circuits With Liquid Cooling on CPU-GPU Platforms. *IEEE Trans. Very Large Scale Integr. Syst.* **2015**, *23*, 575–579.
20. Ouzisik, N. *Heat Transfer—A Basic Approach*; Mcgraw-Hill College Book Company: New York, NY, USA, 1985.
21. Bergman, T.; Lavine, B.; Incropera, P.; DeWitt, P. *Fundamentals of Heat and Mass Transfer*; Wiley: New York, NY, USA, 2017.
22. Barrett, R.; Berry, M.; Chan, T.; Demmel, J.; Donato, J.; Dongarra, J.; Eijkhout, V.; Pozo, R.; Romine, C.; van der Vorst, H. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*, 2nd ed.; SIAM: Philadelphia, PA, USA, 1992.
23. Axelsson, O.; Barker, V.A. *Quadratic Spline Collocation Methods for Elliptic Partial Differential Equations*; Academic Press: Cambridge, MA, USA, 1984.
24. Daloukas, K.; Marnari, A.; Evmorfopoulos, N.; Tsompanopoulou, P.; Stamoulis, G.I. A parallel fast transform-based preconditioning approach for electrical-thermal co-simulation of power delivery networks. In Proceedings of the Design, Automation & Test in Europe Conference & Exhibition (DATE), Grenoble, France, 18–22 March 2013; pp. 1689–1694.
25. Daloukas, K.; Evmorfopoulos, N.; Tsompanopoulou, P.; Stamoulis, G. Parallel Fast Transform-Based Preconditioners for Large-Scale Power Grid Analysis on Graphics Processing Units (GPUs). *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* **2016**, *35*, 1653–1666. [CrossRef]
26. Christara, C.C. Quadratic Spline Collocation Methods for Elliptic Partial Differential Equations. *BIT Numer. Math.* **1994**, *34*, 33–61. [CrossRef]
27. Van Loan, C. *Computational Frameworks for the Fast Fourier Transform*; SIAM: Philadelphia, PA, USA, 1992.
28. NVIDIA CUDA Programming Guide, CUSPARSE, CUBLAS, and CUFFT Library User Guides. Available online: http://developer.nvidia.com/nvidia-gpu-computing-documentation (accessed on 19 December 2018).