Fast Poisson Solvers for Thermal Analysis

HAIFENG QIAN, IBM T. J. Watson Research Center
SACHIN S. SAPATNEKAR, University of Minnesota
EREN KURSUN, IBM T. J. Watson Research Center

Accurate and efficient thermal analysis for a VLSI chip is crucial, both for sign-off reliability verification and for design-time circuit optimization. To determine an accurate temperature profile, it is important to simulate a die together with its thermal mounts: this requires solving Poisson's equation on a nonrectangular 3D domain. This article presents a class of eigendecomposition-based Fast Poisson Solvers (FPS) for chip-level thermal analysis. We start with a solver that solves a rectangular 3D domain with mixed boundary conditions in $O(N \cdot \log N)$ time, where N is the dimension of the finite difference matrix. Then we reveal, for the first time in the literature, a strong relation between fast Poisson solvers and Green-function-based methods. Finally, we propose an FPS method that leverages the preconditioned conjugate gradient method to solve nonrectangular 3D domains efficiently. We demonstrate this approach on thermal analysis of an industrial microprocessor, showing accurate results verified by a commercial tool, and that it solves a system of dimension 4.54e6 in only 13 conjugate gradient iterations, with a runtime of 65 seconds, a 15X speedup over the popular ICCG solver.

1. INTRODUCTION

As power density and cooling costs continue to increase, temperature modeling has become a part of integrated circuit design. Thermal characteristics not only determine the maximum performance envelope of an integrated circuit in the form of thermal design points, they also have strong impact on the overall energy efficiency, cost, and reliability of the design. Since the power dissipation is spatially nonuniform the resulting localized hotspots and thermal gradients create numerous challenges such as clock skews and timing failures. Hotspots also degrade chip reliability over time, leading to hard errors. Temperature-leakage feedback mechanisms imply that hotspots can cause

further power increases and even thermal runaway. Therefore, the ability to obtain an accurate full-chip temperature profile is a necessity during sign-off performance and reliability verification.

Furthermore, accurate and efficient thermal analysis is an indispensable engine for many design-time circuit optimizations. For example, since the power dissipation of a circuit depends on temperature, accurate power analysis and optimization methods must be integrated with thermal analysis. Thermally-driven optimizations require a large number of thermal solves with different power dissipation distributions, and therefore demand extremely high efficiency from the thermal analysis engine.

This article addresses chip-level analysis, solving for the thermal profile of an entire chip. In such an analysis, it is reasonable and common practice to model a die using layered materials, with thermal conductivity being uniform or close to uniform within a layer. This work does not discuss detailed local analysis, for example, using fine-grained models of transistor and metal shapes.

Fast chip-level thermal analysis has been the subject of many prior works [Heriz et al. 2007; Li et al. 2004; Wang and Mazumder 2007; Zhan and Sapatnekar 2007]. Most of them focus on solving Poisson's equation on a die, which is a 3D rectangular domain with layered materials. In a typical model, the boundary conditions on five surfaces of this domain are assumed adiabatic; this is in the form of the *Neumann boundary condition*, which specifies temperature gradient along the direction that is perpendicular to a surface. For adiabatic boundary conditions, this gradient must be zero everywhere on these five surfaces [Li et al. 2004; Zhan and Sapatnekar 2007]. The sixth surface of this domain is assumed convective, and the form that is typically used here is the *Dirichlet boundary condition*, which specifies temperature values on a surface. In this case, the temperature is on the sixth surface of the model [Li et al. 2004; Zhan and Sapatnekar 2007], which is often assumed to be the ambient. A multigrid approach was proposed in Li et al. [2004], which uses finite-difference discretization and a geometric multigrid solver. Several Green-function-based algorithms were proposed in Wang and Mazumder [2007] and Zhan and Sapatnekar [2007], which use the boundary element method and leverage the efficiency of Fast Fourier Transform (FFT). These Green-function-based algorithms have a strong relation to the works of Costa et al. [1999], Gharpurey and Meyer [1996], and Niknejad et al. [1998] on the problems of substrate electrical analysis, which also involves Poisson's equation on a 3D rectangular domain with layered materials.

One important limitation of solving a rectangular domain, the die only, is the simplistic assumption (Dirichlet boundary condition as noted before) about the surface that connects to the thermal mounts. The temperatures on this surface are far from uniform and vary greatly depending on packaging structures. A typical configuration uses a copper heat spreader attached to the die that is wider than the die, and then a copper heat sink attached to the spreader with even larger size and also with fins facing the other side for air cooling [Zhan and Sapatnekar 2007], with thermal interface materials used to ensure good conductivity at the contact surfaces. Although it is possible to approximate these thermal mounts with an effective heat transfer coefficient [Zhan and Sapatnekar 2007], such an approximation may incur substantial error. Therefore, it is important to simulate a die together with its thermal mounts, which requires solving Poisson's equation on a nonrectangular 3D domain [Bagnoli et al. 2007; Heriz et al. 2007].

In fact, it was shown in Heriz et al. [2007] that, with a typical 7cm $\times$ 7cm copper mount under a 1cm $\times$ 1cm silicon die, the on-chip temperatures can be tens of degrees different from a simulation where the mount has the same size as the die, and the shapes of the temperature profiles are also significantly different. Figure 1 illustrates the pyramid-shaped model in Heriz et al. [2007]. This model will be used in Section 5 for
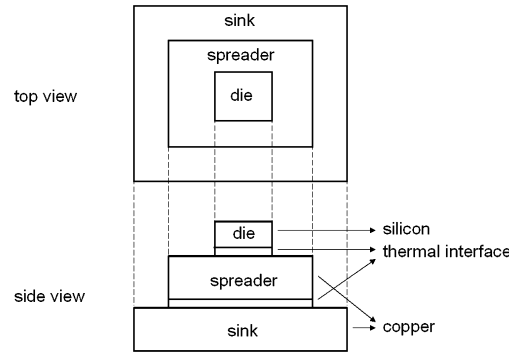
Fig. 1.   The pyramid thermal model from Heriz et al. [2007].

discussion and in Section 6 for testcases; however, our solver in Section 5 is not limited to solving pyramid-shaped domains, and can handle other nonrectangular models, such as those that include fins under the sink.

The difficulty of solving Poisson's equation on a nonrectangular domain is that an analytical form is no longer available for the Green function. Therefore many existing techniques are no longer applicable, and an accurate solution of the finite difference matrix would require generic methods, for example, the finite difference method with the popular Incomplete Cholesky preconditioned Conjugate Gradient (ICCG) solver. As an alternative, Heriz et al. [2007] proposed an imaging technique which precomputes impulse heat-to-temperature responses and then applies convolution to compute temperatures. One drawback of this method is its limited scalability when higher levels of granularity are needed for the power distribution and the temperature profile.

This article studies eigendecomposition-based fast Poisson solvers for chip-level thermal analysis, and throughout this work we will use FPS as a shorthand notation for these solvers. The idea of FPS has been known for a long time [Saad 2003; Swarztrauber 1977], but it has found limited usage in practical applications. In the literature, for example, Saad [2003] and Shi et al. [2009], FPS is often presented in the context of a 2D-grid Laplacian matrix with uniform diagonal entries, which corresponds to a finite difference discretization of a 2D rectangular homogeneous domain with Dirichlet conditions on all four boundaries. It is rarely discussed with respect to 3D domains and/or more general boundary conditions [Swarztrauber 1977].

The major contributions of this article are as follows. Section 3 demonstrates a FPS solver for 3D rectangular thermal models with mixed boundary conditions as in Li et al. [2004] Zhan and Sapatnekar [2007]. The conventional wisdom has been that FPS solvers were an entirely different class of solvers from other known solvers. Section 4 investigates and proves, for the first time, a strong relation (in fact, an equivalence under certain conditions) between FPS and Green-function-based methods. Like Green-function solvers, we show that FPS is also restricted to certain regular structures. Since real circuit structures are not necessarily regular, particularly as related to chip-package interactions, we demonstrate how these regular solvers can be used to devise preconditioned iterative solvers. In particular, Section 5 presents an FPS-Preconditioned Conjugate Gradient method that solves nonrectangular domains efficiently.

2. BASICS: 2D FPS WITH ALL-AROUND DIRICHLET BOUNDARY CONDITIONS

In this section, we review some well-known FPS fundamentals in order to set the stage for introducing more advanced FPS solvers in the rest of this article. Specifically, we

Fig. 2. An example for basic 2D FPS.

discuss using FPS to solve 2D domains with all-around Dirichlet boundary conditions. The discussion is largely based on Saad [2003], and Shi et al. [2009] is an example of applying the method in the design automation domain.

Let us consider a hypothetical 2D problem illustrated in Figure 2. Suppose we use finite difference discretization and divide a 2D domain into $M \cdot K$ cells, and let the x values be the temperatures of each cell; suppose this domain has Dirichlet conditions on all four boundaries, that is, the temperatures of the grey cells are known; suppose the thermal conductivity between any pair of adjacent cells is identical (this is true of a uniform medium) and is equal to g. Then, if we know the power dissipation in each cell, the temperature values can be obtained by solving the linear system.

$$A\mathbf{x} = \mathbf{b}, \text{ where } A = g\begin{pmatrix} B & -I & 0 & 0 \\ -I & B & -I & 0 \\ 0 & -I & \ddots & -I \\ 0 & 0 & -I & B \end{pmatrix}, B = \begin{pmatrix} 4 & -1 & 0 & 0 \\ -1 & 4 & -1 & 0 \\ 0 & -1 & \ddots & -1 \\ 0 & 0 & -1 & 4 \end{pmatrix} \tag{1}$$

Note that the dimension of B is $M \times M$. The eigenvalues λ_i and normalized eigenvectors $\mathbf{q}_i$ of matrix B are well-known [Saad 2003].

$$\forall 1 \leq i \leq M, \quad \lambda_i = 4 - 2\cos\left(\frac{i \cdot \pi}{M+1}\right)$$

$$\mathbf{q_i} = \sqrt{\frac{2}{M+1}} \cdot \left(\sin\left(\frac{i \cdot \pi}{M+1}\right) \quad \sin\left(\frac{2 \cdot i \cdot \pi}{M+1}\right) \quad \cdots \quad \sin\left(\frac{M \cdot i \cdot \pi}{M+1}\right)\right)^{\mathrm{T}} \tag{2}$$

Let matrix Q (which is orthonormal, that is, $Q^T Q = QQ^T = I$) be

$$Q = (\mathbf{q}_1 \quad \mathbf{q}_2 \quad \cdots \quad \mathbf{q}_M). \tag{3}$$

Let us perform the following transformation on (1). We have

$$\begin{pmatrix} Q^{\mathrm{T}} & 0 & 0 & 0 \\ 0 & Q^{\mathrm{T}} & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & Q^{\mathrm{T}} \end{pmatrix} \begin{pmatrix} B & -I & 0 & 0 \\ -I & B & -I & 0 \\ 0 & -I & \ddots & -I \\ 0 & 0 & -I & B \end{pmatrix} \begin{pmatrix} Q & 0 & 0 & 0 \\ 0 & Q & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & Q \end{pmatrix} \tilde{\mathbf{x}} = \tilde{\mathbf{b}}$$

$$\text{where } \tilde{\mathbf{x}} = \begin{pmatrix} Q^{\mathrm{T}} & 0 & 0 & 0 \\ 0 & Q^{\mathrm{T}} & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & Q^{\mathrm{T}} \end{pmatrix} \mathbf{x}, \tilde{\mathbf{b}} = \frac{1}{g} \begin{pmatrix} Q^{\mathrm{T}} & 0 & 0 & 0 \\ 0 & Q^{\mathrm{T}} & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & Q^{\mathrm{T}} \end{pmatrix} \mathbf{b}, \tag{4}$$

which becomes

$$\begin{pmatrix} \Lambda & -I & 0 & 0 \\ -I & \Lambda & -I & 0 \\ 0 & -I & \ddots & I \\ 0 & 0 & I & \Lambda \end{pmatrix} \tilde{\mathbf{x}} = \tilde{\mathbf{b}}, \tag{5}$$

where Λ is a diagonal matrix with diagonal entries being $\lambda_1, \lambda_2, \ldots, \lambda_M$. Because of the block form of (5) and each subblock being a diagonal matrix, (5) can be decoupled into M independent systems, each with a left-hand-side matrix in the following form.

$$\begin{pmatrix} \lambda_i & -1 & 0 & 0 \\ -1 & \lambda_i & -1 & 0 \\ 0 & -1 & \ddots & -1 \\ 0 & 0 & -1 & \lambda_i \end{pmatrix}, \quad 1 \le i \le M \tag{6}$$

The preceding is a tridiagonal matrix, and each such subsystem can be solved in linear time [Saad 2003]. After (5) is solved, the solution to the original system is simply

$$\mathbf{x} = \begin{pmatrix} Q & 0 & 0 & 0 \\ 0 & Q & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & Q \end{pmatrix} \tilde{\mathbf{x}}. \tag{7}$$

The efficiency of this FPS comes from the fact that both the computation of $\tilde{\mathbf{b}}$ from $\mathbf{b}$ by (4), and the computation of $\mathbf{x}$ from $\tilde{\mathbf{x}}$ by (7), can be done using FFT with $O(K \cdot M \cdot \log M)$ complexity.

3. 3D FPS WITH MIXED BOUNDARY CONDITIONS

The previous section reviews the known solution to a 2D rectangular homogeneous domain with Dirichlet conditions on all boundaries. It represents a very restrictive scenario where the matrix takes the form of (1). In this section, we expand the applicability of FPS.

3.1. Principles

Let us present the theory in the scenario with Neumann boundary conditions on five surfaces and Dirichlet boundary condition on the last one, which is a typical model used for chip-level thermal analysis [Li et al. 2004; Zhan and Sapatnekar 2007]. Implications of other boundary conditions and transient analysis will be discussed in Section 3.2.

The following is a relaxed sufficient condition for FPS.

$$A = \begin{pmatrix} B_{1,1} & B_{1,2} & \cdots & B_{1,K} \\ B_{2,1} & B_{2,2} & \cdots & B_{2,K} \\ \vdots & \vdots & \ddots & \vdots \\ B_{K,1} & B_{K,2} & \cdots & B_{K,K} \end{pmatrix} \tag{8}$$

such that
$B_{i,j}, \ \forall i, j$ are square matrices with the same dimension;
$B_{i,j}, \ \forall i, j$ have the same eigenvectors.

Let the common normalized eigenvectors of $B_{i,j}$ be $\mathbf{q}_1$, $\mathbf{q}_2$, ... , $\mathbf{q}_M$, where M is the dimension of $B_{i,j}$. Let matrix Q be as in (3), but with these eigenvectors of $B_{i,j}$.

To solve $A\mathbf{x} = \mathbf{b}$, let us do the following transformation, as in (4):

$$
\begin{pmatrix} Q^{\mathrm{T}} & 0 & 0 & 0 \\ 0 & Q^{\mathrm{T}} & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & Q^{\mathrm{T}} \end{pmatrix} A \begin{pmatrix} Q & 0 & 0 & 0 \\ 0 & Q & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & Q \end{pmatrix} \begin{pmatrix} Q^{\mathrm{T}} & 0 & 0 & 0 \\ 0 & Q^{\mathrm{T}} & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & Q^{\mathrm{T}} \end{pmatrix} \mathbf{x} = \begin{pmatrix} Q^{\mathrm{T}} & 0 & 0 & 0 \\ 0 & Q^{\mathrm{T}} & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & Q^{\mathrm{T}} \end{pmatrix} \mathbf{b}
$$

$$
\begin{pmatrix} \Lambda_{1,1} & \Lambda_{1,2} & \cdots & \Lambda_{1,K} \\ \Lambda_{2,1} & \Lambda_{2,2} & \cdots & \Lambda_{2,K} \\ \vdots & \vdots & \ddots & \vdots \\ \Lambda_{K,1} & \Lambda_{K,2} & \cdots & \Lambda_{K,K} \end{pmatrix} \cdot \tilde{\mathbf{x}} = \tilde{\mathbf{b}} \tag{9}
$$

where $\tilde{\mathbf{x}} = \begin{pmatrix} Q^{\mathrm{T}} & 0 & 0 & 0 \\ 0 & Q^{\mathrm{T}} & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & Q^{\mathrm{T}} \end{pmatrix} \mathbf{x}$, $\tilde{\mathbf{b}} = \begin{pmatrix} Q^{\mathrm{T}} & 0 & 0 & 0 \\ 0 & Q^{\mathrm{T}} & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & Q^{\mathrm{T}} \end{pmatrix} \mathbf{b}$

where $\Lambda_{i,j}$ is a diagonal matrix with diagonal entries being the eigenvalues of $B_{i,j}$.

Now that the new left-hand-side matrix has a block form where each block is a diagonal submatrix, the overall system of linear equations can be decoupled into M separate sets of linear equations, each of which has dimension K and can be solved independently. Note that, if these sets of equations also satisfy condition (8), each of them can again be transformed and further decoupled into smaller sets of equations. Again, after $\tilde{\mathbf{x}}$ is obtained, solution $\mathbf{x}$ to the original system can be obtained by (7).

The advantage of FPS comes when Q has a special structure such that a fast way exists to evaluate $\tilde{\mathbf{b}}$ from $\mathbf{b}$ by (9), and evaluate $\mathbf{x}$ from $\tilde{\mathbf{x}}$ by (7).

Now let us apply the aforesaid principles on a 3D thermal analysis model with Neumann boundary conditions on five surfaces, and the Dirichlet boundary condition on the $z = 0$ surface [Li et al. 2004; Zhan and Sapatnekar 2007].

Let a and b be the dimensions of the domain in x and y directions. We assume even-spaced discretization along the x-axis into D_x segments, even-spaced discretization along the y-axis into D_y segments, and arbitrary discretization along the z-axis into D_z segments. The result is $D_x \cdot D_y \cdot D_z$ rectangular cells. For $0 \le m \le D_x - 1$, $0 \le n \le D_y - 1$, $0 \le l \le D_z - 1$, we use the notion cell (m,n,l) to refer to the cell formed by the mth segment of the x axis, the nth segment of the y axis, and the lth segment of the z axis. Let $T_{m,n,l}$ be the average temperature of cell (m,n,l), and let $P_{m,n,l}$ be the total power dissipation inside cell (m, n, l). We assume a natural ordering in the linear system $A\mathbf{x} = \mathbf{b}$, in other words,

$$
x_{m+n\cdot D_x+l\cdot D_x \cdot D_y} = T_{m,n,l} - T_a, \tag{10}
$$

$$
b_{m+n\cdot D_x+l\cdot D_x \cdot D_y} = P_{m,n,l}, \tag{11}
$$

where T_a is the ambient temperature at the external surface of the heat sink.

Applying Eq. (8) with $M = D_x$ and $K = D_y \cdot D_z$, we have special forms for the $B_{i,j}$ matrices.

$$\forall i, \quad B_{i,i} = \alpha_i \cdot I + \beta_i \cdot G$$

where α_i and β_i are scalars, $G = \begin{pmatrix} -1 & -1 & 0 & \cdots & 0 & 0 & 0 \\ -1 & 0 & -1 & \cdots & 0 & 0 & 0 \\ 0 & -1 & 0 & \ddots & 0 & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \ddots & 0 & -1 & 0 \\ 0 & 0 & 0 & \cdots & -1 & 0 & -1 \\ 0 & 0 & 0 & \cdots & 0 & -1 & -1 \end{pmatrix}$ (12)

$\forall i \neq j, \ B_{i,j} = \gamma_{i,j} \cdot I$
where $\gamma_{i,j}$ is a scalar.

Note that the $B_{i,i}$ formulation in the previous equation is *different* from (1) and does not subsume (1). This is due to the fact that we now have Neumann conditions on the $x = 0$ and $x = a$ surfaces, as opposed to Dirichlet conditions in Section 2. Consequently, the eigenvectors of $B_{i,i}$ matrices in (12), which are essentially eigenvectors of G, must be different from (2).

The eigenvalues and eigenvectors of G also have analytical forms. It can be verified that they are

$$\forall 1 \leq i \leq M, \quad \lambda_i = -2\cos\left(\frac{(i-1) \cdot \pi}{M}\right)$$

$$\mathbf{q}_1 = \sqrt{\frac{1}{M}} \cdot \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix} \qquad \forall i > 1, \mathbf{q}_i = \sqrt{\frac{2}{M}} \cdot \begin{pmatrix} \cos\left(\frac{(i-1)\cdot\pi}{2M}\right) \\ \cos\left(\frac{3\cdot(i-1)\cdot\pi}{2M}\right) \\ \vdots \\ \cos\left(\frac{(2M-1)\cdot(i-1)\cdot\pi}{2M}\right) \end{pmatrix}.$$ (13)

Let matrix Q be defined as in (3), but using the preceding eigenvectors instead. Fortunately, with these vectors, the computation of $Q^{\mathrm{T}}\mathbf{v}$ and $Q\mathbf{v}$, for any vector $\mathbf{v}$, can also be performed by FFT and with $O(M \cdot \log M)$ complexity.

Now we have details of the transformed system in (9).

$$\forall i, \qquad \Lambda_{i,i} = Q^{\mathrm{T}} B_{i,i} Q = \alpha_i \cdot I + \beta_i \cdot \begin{pmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_M \end{pmatrix}$$ (14)

$$\forall i \neq j, \quad \Lambda_{i,j} = Q^{\mathrm{T}} B_{i,j} Q = \gamma_{i,j} \cdot I$$

Let us investigate the complexity of FPS on this thermal analysis. The cost of computing $\tilde{\mathbf{b}}$ from $\mathbf{b}$ by Eq. (9), using FFT, is $O(K \cdot M \cdot \log M) = O(D_x \cdot D_y \cdot D_z \cdot \log D_x)$. The cost of computing the final solution $\mathbf{x}$ by Eq. (7), using FFT, is also $O(D_x \cdot D_y \cdot D_z \cdot \log D_x)$. The remaining question is the cost of the middle step, solving Eq. (9) as D_x separate systems, each with dimension $D_y \cdot D_z$.

Studying Eqs. (9)(12)(14) in more details, it can be shown that each of these $D_y \cdot D_z$ matrices has a 2D *y-z* grid structure and also satisfies conditions (8)(12).

Therefore, we can apply the FPS procedure on each of them on the y direction (note that we have Neumann conditions on $y = 0$ and $y = b$ surfaces). It again is a three-step approach: the first and third steps use the FFT, with complexity $O(D_y \cdot D_z \cdot \log D_y)$, and the middle step now solves D_y separate systems, each with dimension D_z. Each of them is now a 1D structure in the z direction, which is a tridiagonal matrix and can be solved in linear time. Therefore, the cost of solving a system with dimension $D_y \cdot D_z$ is $O(D_y \cdot D_z \cdot \log D_y) + O(D_y \cdot D_z) + O(D_y \cdot D_z \cdot \log D_y)$, which is bounded by $O(D_y \cdot D_z \cdot \log D_y)$.

Therefore, the overall cost of solving the original system $A\mathbf{x} = \mathbf{b}$ is $O(D_x \cdot D_y \cdot D_z \cdot \log D_x) + D_x \cdot O(D_y \cdot D_z \cdot \log D_y)$, which is bounded by $O(N \cdot \log N)$, where $N = D_x \cdot D_y \cdot D_z$ is the overall dimension.

3.2. Other Thermal Models and Transient Analysis

The previous section covers the most widely used thermal model. This section discusses other thermal models, in particular the implications on FPS of various boundary conditions, as well as transient thermal analysis.

One possible thermal model has Neumann boundary conditions on five surfaces as in the previous section, but has a *Robin boundary condition* instead of Dirichlet boundary condition on the sixth surface. The *Robin boundary condition* is a linear combination of Dirichlet and Neumann boundary conditions, and specifies a weighted sum of the temperature and the perpendicular temperature gradient at every point on the surface. Typically we can assume uniform material property across the surface, that is, that the two weighting coefficients are constant. Under finite difference discretization, the Robin boundary condition's implication to our linear system (8)(12) is that α_i's in (12) are modified for i's that correspond to rows on the $z = 0$ surface. These changes remain only on diagonals in the transformed system (9)(14), similarly remain after the FPS transformation on y direction, and eventually only affect the tridiagonal systems in the z direction. Therefore the entire algorithm in Section 3.1 applies and the complexity remains $O(N \cdot \log N)$.

More complex thermal models have been applied on chips with microchannel cooling [Mizunuma et al. 2009; Sridhar et al. 2010; Sabry et al. 2011]. Without loss of generality, let us assume that the coolant flows from the $y = 0$ surface to the $y = b$ surface. In the finite difference matrix (also referred to as equivalent resistive network in these papers), there are now variations in thermal conductivity along x and y directions at the microchannel layers. The variation along the x direction can be handled by the technique to be presented in Section 5, but the variation along the y direction is significant, particularly with extra elements in Mizunuma et al. [2009] to model the thermal-wakes effect. In addition, the boundary condition on the $y = 0$ surface becomes partially Dirichlet (where the coolant inlets are) and partially Neumann [Sridhar et al. 2010]. With such thermal models, FPS can only perform the first round of transformation along the x direction, and not the second round along the y direction. Consequently, as shown in the previous section, the overall complexity is $O(D_x \cdot D_y \cdot D_z \cdot \log D_x)$ plus solving Eq. (9) as D_x separate systems, each with a 2D grid structure and with dimension $D_y \cdot D_z$. The complexity of solving each 2D grid with a direct solver with nested-dissection ordering such as METIS [Karypis et al. 1995] is $O(D_y^{1.5} \cdot D_z^{1.5})$. Therefore the complexity bound is $O(D_x \cdot D_y \cdot D_z \cdot \log D_x + D_x \cdot D_y^{1.5} \cdot D_z^{1.5})$, which is still more efficient than solving the original 3D system as a whole.

The proposed FPS method is applicable to transient thermal analysis. A common practice is to use an equivalent thermal RC network, which includes grounded capacitors [Huang et al. 2006] to represent heat storage inside each cell per unit increase of

temperature. A transient analysis is thus to solve

$$A\mathbf{x}(t) + C\frac{d\mathbf{x}(t)}{dt} = \mathbf{b}(t),$$

where A is the same thermal conductance matrix as in the previous session, C is a diagonal matrix with thermal capacitance values, and $\mathbf{b}(t)$ is time-varying power distribution. With the Backward Euler method (other methods follow similar derivations) with time step size h, the analysis becomes solving the following linear system at each time step.

$$\left(A + \frac{C}{h}\right)\mathbf{x}(t) = \mathbf{b}(t) + \frac{C\mathbf{x}(t-h)}{h}.$$

It is important to note that, since C is a diagonal matrix, the left-hand-side matrix is only modified at the diagonal entries, and that, assuming layered materials and hence uniform thermal capacitance per layer, the new left-hand-side matrix still takes the form of (8)(12). Therefore the entire algorithm in Section 3.1 applies and the complexity is $O(N \cdot \log N)$ per time step.

As a last note, a few other boundary conditions, though in 2D domains, were discussed in Swarztrauber [1977]. Their formulation for the Neumann condition is different from (12). The difference comes from a different approach of sampling derivatives under finite difference discretization. Both formulations are correct, and both are within the approximation introduced by the finite difference method.

4. RELATION BETWEEN FPS AND GREEN-FUNCTION-BASED METHODS

This section demonstrates a strong relation—in fact equivalence under certain conditions—between the 3D FPS from Section 3 and Green-function-based Poisson solvers [Gharpurey and Meyer 1996; Niknejad et al. 1998; Zhan and Sapatnekar 2007]. For clarity of presentation, we will only compare with the full-chip thermal analysis method, referred to as Algorithm II in Zhan and Sapatnekar [2007]. The relation to other variations of Green-function-based methods follows similar derivations. For the rest of the article, we will use GFS as a short-hand notation for Green-function-based solvers.

For clarity and without loss of generality, we assume, as in Zhan and Sapatnekar [2007], that heat sources are located on discrete horizontal planes. Also, like Zhan and Sapatnekar [2007], we assume that each of these planes is divided into $D_x \times D_y$ rectangular grid cells of equal size, and the power density inside each grid cell is uniform. Therefore, the power density distribution function, P, is in the following piecewise constant form, which is nonzero only at a finite set of z' values. We have

$$P(x', y', z') = \sum_{m=0}^{D_x-1}\sum_{n=0}^{D_y-1} P_{m,n}(z') \cdot \Theta\left(x' - (m+0.5)\frac{a}{D_x}, y' - (n+0.5)\frac{b}{D_y}\right)$$

where   (15)

$$\Theta(x', y') = \begin{cases} 1, & |x'| < \dfrac{a}{2D_x}, |y'| < \dfrac{b}{2D_y} \\ 0, & \text{otherwise} \end{cases}$$

where again a and b are the dimensions of the domain in x and y directions. Note that the discreteness assumption about z axis is not essential, and, on removing it, the derivation still stands, only with constants $P_{m,n}(z')$ replaced by more complex forms.

4.1. FPS Formulations

This section gives detailed FPS formulations based on the principles of Section 3, such that they can be compared with GFS formulations that will be given in Section 4.2.

Let us use the same formation of linear system $A\mathbf{x} = \mathbf{b}$ as (10)(11), and assume that the x and y discretization is the same as in (15). Let the z coordinates of cell centers be $z_0, z_1, \ldots, z_{D_z-1}$. For clarity of presentation, we further assume that heat source planes in (15) coincide with these z coordinates. In other words,

$$P_{m,n}\left(z'\right) = 0, \quad P(x', y', z') = 0, \quad \text{for } z' \notin \{z_1, z_2, \cdots z_{D_z-1}\}. \tag{16}$$

And we have the following relation between (11) and (15).

$$b_{m+n\cdot D_x+l\cdot D_x\cdot D_y} = P_{m,n,l} = P_{m,n}(z_l) \cdot \frac{a \cdot b}{D_x \cdot D_y} \tag{17}$$

The first step of FPS is to apply the transformation of (9) with $M = D_x$ and $K = D_y \cdot D_z$, and compute the transformed right-hand side. We have

$$\tilde{\mathbf{b}} = \begin{pmatrix} Q_{D_x}^{\mathrm{T}} & 0 & 0 & 0 \\ 0 & Q_{D_x}^{\mathrm{T}} & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & Q_{D_x}^{\mathrm{T}} \end{pmatrix} \mathbf{b}, \tag{18}$$

where $Q_{D_x}^{\mathrm{T}}$ denotes the eigenvector matrix from (3)(13) with dimension $M = D_x$.

The new system (9) is then decoupled into D_x independent systems of linear equations. The ith system is, for $0 \le i \le D_x - 1$

$$\tilde{A}_i \tilde{\mathbf{x}}_{\mathbf{i}} = \tilde{\mathbf{b}}_{\mathbf{i}} \tag{19}$$

where $\tilde{A}_i$ is a submatrix of the transformed left-hand side in (9), $\tilde{\mathbf{x}}_{\mathbf{i}}$ is a part of $\tilde{\mathbf{x}}$, and $\tilde{\mathbf{b}}_{\mathbf{i}}$ is a part of $\tilde{\mathbf{b}}$, such that

$$\tilde{x}_{i,r} = \tilde{x}_{i+D_x\cdot r}, \tilde{b}_{i,r} = \tilde{b}_{i+D_x\cdot r}, \forall 0 \le r \le D_y \cdot D_z - 1. \tag{20}$$

Then we apply the transformation of (9) on (19) with $M = D_y$ and $K = D_z$, and compute another transformed right-hand side

$$\tilde{\tilde{\mathbf{b}}}_{\mathbf{i}} = \begin{pmatrix} Q_{D_y}^{\mathrm{T}} & 0 & 0 & 0 \\ 0 & Q_{D_y}^{\mathrm{T}} & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & Q_{D_y}^{\mathrm{T}} \end{pmatrix} \tilde{\mathbf{b}}_{\mathbf{i}}, \tag{21}$$

where $Q_{D_y}^{\mathrm{T}}$ denotes the eigenvector matrix from (3)(13) with dimension $M = D_y$. And (19) now becomes

$$
\begin{pmatrix} Q_{D_y}^{\mathrm{T}} & 0 & 0 & 0 \\ 0 & Q_{D_y}^{\mathrm{T}} & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & Q_{D_y}^{\mathrm{T}} \end{pmatrix} \tilde{A}_i \begin{pmatrix} Q_{D_y} & 0 & 0 & 0 \\ 0 & Q_{D_y} & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & Q_{D_y} \end{pmatrix} \tilde{\tilde{\mathbf{x}}}_i = \tilde{\tilde{\mathbf{b}}}_i
$$

$$
\text{where } \tilde{\tilde{\mathbf{x}}}_i = \begin{pmatrix} Q_{D_y}^{\mathrm{T}} & 0 & 0 & 0 \\ 0 & Q_{D_y}^{\mathrm{T}} & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & Q_{D_y}^{\mathrm{T}} \end{pmatrix} \tilde{\mathbf{x}}_i.
\tag{22}
$$

Vector $\tilde{\tilde{\mathbf{x}}}_i$ can be solved from the preceding equation in $O(D_y \cdot D_z)$ time because it can be decoupled into D_y independent systems, each of which is a tridiagonal matrix of dimension D_z.

After obtaining $\tilde{\tilde{\mathbf{x}}}_i$, we then compute

$$
\tilde{\mathbf{x}}_i = \begin{pmatrix} Q_{D_y} & 0 & 0 & 0 \\ 0 & Q_{D_y} & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & Q_{D_y} \end{pmatrix} \tilde{\tilde{\mathbf{x}}}_i.
\tag{23}
$$

After having $\tilde{\mathbf{x}}_i$ for $0 \le i \le D_x - 1$, we can now recover $\tilde{\mathbf{x}}$ based on Eq. (20). Then the solution to the original system is

$$
\mathbf{x} = \begin{pmatrix} Q_{D_x} & 0 & 0 & 0 \\ 0 & Q_{D_x} & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & Q_{D_x} \end{pmatrix} \tilde{\mathbf{x}}.
\tag{24}
$$

Finally, the temperature values at each cell are mapped from (10).

Overall, the FPS procedure can be summarized as follows.
1. Calculate vector $\tilde{\mathbf{b}}$ from $P_{m,n}(z')$ by equations (17)(18), using FFT.
2. Calculate vectors $\tilde{\tilde{\mathbf{b}}}_i$ from $\tilde{\mathbf{b}}$ by equations (20)(21), for $0 \le i \le D_x - 1$, using FFT.
3. Calculate vectors $\tilde{\tilde{\mathbf{x}}}_i$ by solving (22) as D_y independent linear systems, each with dimension D_z, for $0 \le i \le D_x - 1$.
4. Calculate vector $\tilde{\mathbf{x}}$ from $\tilde{\tilde{\mathbf{x}}}_i$, $0 \le i \le D_x - 1$, by equations (20)(23), using FFT.
5. Calculate vector $\mathbf{x}$ from $\tilde{\mathbf{x}}$ by equation (24), using FFT.

To study the relation to GFS, let us expand the equations of the first two steps. Symbol $\mathbf{q}_{M,i}$ will be used to denote the ith eigenvector in Eq. (13) with dimension M. By Eqs. (13)(17)(18)(20)(21), we have

$$
\text{for } 0 \le i \le D_x - 1, 0 \le j \le D_y - 1, 0 \le l \le D_z - 1
$$

$$
\tilde{b}_{i,j+D_y \cdot l} = \left(\mathbf{q}_{D_y, j+1} \right)^{\mathrm{T}} \cdot \left(\breve{b}_{i,D_y \cdot l} \quad \breve{b}_{i,D_y \cdot l+1} \cdots \breve{b}_{i,D_y \cdot l+D_y-1} \right)^{\mathrm{T}}
$$

$$
= \chi_j \cdot \sqrt{\frac{1}{D_y}} \cdot \sum_{n=0}^{D_y-1} \cos \frac{j\pi(n+0.5)}{D_y} \cdot \breve{b}_{i,D_y \cdot l+n}
$$

$$
= \chi_j \cdot \sqrt{\frac{1}{D_y}} \cdot \sum_{n=0}^{D_y-1} \cos \frac{j\pi(n+0.5)}{D_y} \cdot \tilde{b}_{i+D_x \cdot n + D_x \cdot D_y \cdot l}
$$

$$
= \chi_j \cdot \sqrt{\frac{1}{D_y}} \cdot \sum_{n=0}^{D_y-1} \cos \frac{j\pi(n+0.5)}{D_y} \cdot \left(\mathbf{q}_{D_x,l+1}\right)^{\mathrm{T}} \cdot \begin{pmatrix} b_{D_x \cdot n + D_x \cdot D_y \cdot l} \\ b_{D_x \cdot n + D_x \cdot D_y \cdot l + 1} \\ \vdots \\ b_{D_x \cdot n + D_x \cdot D_y \cdot l + D_x - 1} \end{pmatrix}
$$

$$
= \chi_i \cdot \chi_j \cdot \sqrt{\frac{1}{D_x D_y}} \cdot \sum_{m=0}^{D_x-1} \sum_{n=0}^{D_y-1} P_{m,n,j} \cdot \cos \frac{i\pi(m+0.5)}{D_x} \cdot \cos \frac{j\pi(n+0.5)}{D_y} \quad (25)
$$

$$
= \frac{\chi_i \cdot \chi_j \cdot a \cdot b}{D_x^{1.5} \cdot D_y^{1.5}} \cdot \sum_{m=0}^{D_x-1} \sum_{n=0}^{D_y-1} P_{m,n}(z_l) \cdot \cos \frac{i\pi(m+0.5)}{D_x} \cdot \cos \frac{j\pi(n+0.5)}{D_y}
$$

$$
\text{where } \chi_j = \begin{cases} 1, & j = 0 \\ \sqrt{2}, & j > 0. \end{cases}
$$

Similarly, we can expand the last two steps of FPS, and by (10)(13)(20)(23)(24), the average temperature of cell (m,n,l) is

$$
\text{for } 0 \le m \le D_x - 1, 0 \le n \le D_y - 1, 0 \le l \le D_z - 1
$$

$$
T_{m,n,l} = T_a + x_{m+n \cdot D_x + l \cdot D_x \cdot D_y}
$$

$$
= T_a + \sqrt{\frac{1}{D_x}} \cdot \sum_{i=0}^{D_x-1} \chi_i \cdot \cos \frac{i\pi(m+0.5)}{D_x} \cdot \tilde{x}_{i+n \cdot D_x + l \cdot D_x \cdot D_y} \quad (26)
$$

$$
= T_a + \sqrt{\frac{1}{D_x}} \cdot \sum_{i=0}^{D_x-1} \chi_i \cdot \cos \frac{i\pi(m+0.5)}{D_x} \cdot \tilde{x}_{i,n \cdot + l \cdot D_y}
$$

$$
= T_a + \sqrt{\frac{1}{D_x D_y}} \cdot \sum_{i=0}^{D_x-1} \sum_{j=0}^{D_y-1} \chi_i \cdot \chi_j \cdot \tilde{\tilde{x}}_{i,j+l \cdot D_y} \cdot \cos \frac{i\pi(m+0.5)}{D_x} \cdot \cos \frac{j\pi(n+0.5)}{D_y}
$$

where χ_j is as defined in (25).

4.2. GFS Formulations

This section is largely based on Algorithm II in Zhan and Sapatnekar [2007]. The GFS approach evaluates the following temperature calculation formula. We have

$$
T(x,y,z) = T_a + \sum_{z'} \int_0^a dx' \int_0^b dy' G(x,y,z,x',y',z') P(x',y',z'), \quad (27)
$$

where $G(.)$ is the Green function. In the special case of Neumann conditions on 4 side surfaces ($x = 0$, $y = 0$, $x = a$, $y = b$), according to Niknejad et al. [1998] Zhan and Sapatnekar [2007], it has the following special form.

$$
G(x,y,z,x',y',z') = \sum_{i=0}^{\infty} \sum_{j=0}^{\infty} C_{i,j}(z,z') \cdot \cos \frac{i\pi x}{a} \cdot \cos \frac{j\pi y}{b} \cdot \cos \frac{i\pi x'}{a} \cdot \cos \frac{j\pi y'}{b} \quad (28)
$$

Niknejad et al. [1998] provided ways to calculate function $C_{i,j}(.)$ analytically.

Now suppose we write the power density distribution (15) in the following frequency-domain form.

$$P(x', y', z') = \sum_{i=0}^{\infty} \sum_{j=0}^{\infty} p_{i,j}\left(z'\right) \cdot \cos \frac{i\pi x'}{a} \cdot \cos \frac{j\pi y'}{b} \tag{29}$$

By (15) and (29), we can derive

$$p_{i,j}(z') = a \cdot b \cdot \mu_{i,j} \cdot \sum_{m=0}^{D_x-1} \sum_{n=0}^{D_y-1} P_{m,n}(z') \cdot \cos \frac{i\pi(m+0.5)}{D_x} \cdot \cos \frac{j\pi(n+0.5)}{D_y}$$

$$\text{where } \mu_{i,j} = \begin{cases} \dfrac{1}{D_x D_y}, & i = j = 0 \\[2mm] \dfrac{2}{j\pi D_x} \sin \dfrac{j\pi}{2D_y}, & i = 0, j \neq 0 \\[2mm] \dfrac{2}{i\pi D_y} \sin \dfrac{i\pi}{2D_x}, & i \neq 0, j = 0 \\[2mm] \dfrac{4}{ij\pi^2} \sin \dfrac{i\pi}{2D_x} \sin \dfrac{j\pi}{2D_y}, & i \neq 0, j \neq 0 \end{cases} \tag{30}$$

Substituting (28) and (29) into (27), we get an alternative to (27).

$$T\left(x, y, z\right) = T_a + a \cdot b \cdot \sum_{z'} \sum_{i=0}^{\infty} \sum_{j=0}^{\infty} \omega_{i,j} \cdot p_{i,j}\left(z'\right) \cdot C_{i,j}\left(z, z'\right) \cdot \cos \frac{i\pi x}{a} \cdot \cos \frac{j\pi y}{b}$$

$$\text{where } \omega_{i,j} = \begin{cases} 1, & i = j = 0 \\ 0.5, & i = 0, j \neq 0 \\ 0.5, & i \neq 0, j = 0 \\ 0.25, & i \neq 0, j \neq 0 \end{cases} \tag{31}$$

Now define

$$Z_{i,j}\left(z\right) = \sum_{z'} \omega_{i,j} \cdot p_{i,j}\left(z'\right) \cdot C_{i,j}\left(z, z'\right) \tag{32}$$

and we have a cleaner look of (31).

$$T\left(x, y, z\right) = T_a + a \cdot b \cdot \sum_{i=0}^{\infty} \sum_{j=0}^{\infty} Z_{i,j}\left(z\right) \cdot \cos \frac{i\pi x}{a} \cdot \cos \frac{j\pi y}{b} \tag{33}$$

Practically, we need to truncate the infinite summations in (33). By truncating the two infinite sums at D'_x and D'_y respectively,

$$T\left(x, y, z\right) = T_a + a \cdot b \cdot \sum_{i=0}^{D'_x-1} \sum_{j=0}^{D'_y-1} Z_{i,j}\left(z\right) \cdot \cos \frac{i\pi x}{a} \cdot \cos \frac{j\pi y}{b}. \tag{34}$$

Also in practice, as is done in Zhan and Sapatnekar [2007], often we are interested in not the direct outcome of (34), but, by dividing the x-y plane into evenly D''_x by D''_y rectangular cells, the average temperature in each cell. They are the following numbers.

for $0 \leq m \leq D_x'' - 1, 0 \leq n \leq D_y'' - 1$

$$T_{m,n}(z) = \frac{D_x'' D_y''}{ab} \int_{x=\frac{am}{D_x''}}^{\frac{a(m+1)}{D_x''}} dx \int_{y=\frac{bn}{D_y''}}^{\frac{b(n+1)}{D_y''}} dy T(x,y,z)$$

$$= T_a + a \cdot b \cdot D_x'' \cdot D_y'' \cdot \sum_{i=0}^{D_x'-1} \sum_{j=0}^{D_y'-1} \theta_{i,j} \cdot Z_{i,j}(z) \cdot \cos \frac{i\pi(m+0.5)}{D_x''} \cdot \cos \frac{j\pi(n+0.5)}{D_y''}$$

$$\text{where } \theta_{i,j} = \begin{cases} \dfrac{1}{D_x'' D_y''}, & i = j = 0 \\[2mm] \dfrac{2}{j\pi D_x''} \sin \dfrac{j\pi}{2 D_y''}, & i = 0, j \neq 0 \\[2mm] \dfrac{2}{i\pi D_y''} \sin \dfrac{i\pi}{2 D_x''}, & i \neq 0, j = 0 \\[2mm] \dfrac{4}{ij\pi^2} \sin \dfrac{i\pi}{2 D_x''} \sin \dfrac{j\pi}{2 D_y''}, & i \neq 0, j \neq 0 \end{cases} \tag{35}$$

Overall, the GFS procedure can be summarized as follows.
1. Calculate function $C_{i,j}(z,z')$ for $0 \leq i \leq D_x' - 1$, $0 \leq j \leq D_y' - 1$, analytically as in [8].
2. Calculate function $p_{i,j}(z')$ by (30), for $0 \leq i \leq D_x' - 1, 0 \leq j \leq D_y' - 1$, using FFT.
3. Calculate function $Z_{i,j}(z)$ by (32), for $0 \leq i \leq D_x' - 1, 0 \leq j \leq D_y' - 1$.
4. Calculate function $T_{m,n}(z)$ by (35), for $0 \leq m \leq D_x'' - 1, 0 \leq n \leq D_y'' - 1$, using FFT.

4.3. Relating FPS to GFS

Let us consider the special case of GFS where $D_x = D_x' = D_x''$ and $D_y = D_y' = D_y''$. It is true that GFS is more flexible than FPS in the sense that it has three resolution controls: the resolution (D_x, D_y) of the heat source distribution, the resolution (D_x', D_y') of frequency-domain truncation, and the resolution (D_x'', D_y'') of temperature output. However, these three are not completely independent choices, for example, it was shown in Zhan and Sapatnekar [2007] that D_x' should be a multiple of D_x and D_y' should be a multiple of D_y, to enable efficient implementation.

It is easy to see the relation between Eq. (25), which is steps 1 and 2 of FPS, and Eq. (30), which is step 2 of GFS. Specifically, both equations represent the same computation and produce values merely differing by a scaling factor.

$$\text{for } 0 \leq i \leq D_x - 1, 0 \leq j \leq D_y - 1, 0 \leq l \leq D_z - 1$$
$$p_{i,j}(z_l) \equiv \frac{\mu_{i,j} \cdot D_x^{1.5} \cdot D_y^{1.5}}{\chi_i \cdot \chi_j} \cdot \tilde{b}_{i,j+D_y \cdot l} \tag{36}$$

Similarly, there is a correspondence between (26), which is steps 4 and 5 of FPS, and Eq. (35), which is step 4 of GFS. Specifically, both equations represent the same computation and produce the same temperature outcome (subject to numerical errors),

based on input values $\tilde{\tilde{x}}_\mathbf{i}$ and $Z_{i,j}(z)$ respectively, which merely differ by a scaling factor.

$$\text{for } 0 \leq i \leq D_x - 1,\, 0 \leq j \leq D_y - 1,\, 0 \leq l \leq D_z - 1$$

$$T_{m,n}(z_l) \approx T_{m,n,l} \tag{37}$$

$$Z_{i,j}(z_l) \approx \frac{\chi_i \cdot \chi_j}{a \cdot b \cdot D_x^{1.5} \cdot D_y^{1.5} \cdot \theta_{i,j}} \cdot \tilde{\tilde{x}}_{i,j+l \cdot D_y}$$

Note that the preceding is not an exact equality, because $\tilde{\tilde{x}}_\mathbf{i}$ and $Z_{i,j}(z)$ are obtained in different ways. Specifically, step 3 of FPS computes $\tilde{\tilde{x}}_\mathbf{i}$ by solving $D_x \cdot D_y$ independent linear systems, while steps 1 and 3 of GFS obtain $Z_{i,j}(z)$ by solving the z-direction analytically. Although through different approaches, step 3 of FPS and steps 1 and 3 of GFS essentially do the same task: they take the same input, as shown in (36), and produce approximately the same output, as shown in (37).

In conclusion, FPS and GFS are mathematically equivalent for the special case of $D_x = D_x' = D_x''$ and $D_y = D_y' = D_y''$. In other words, the Green-function-based approach with frequency-domain truncation is equivalent to even-spaced finite difference discretization. This is not surprising, because it is well-known in signal processing theory that frequency-domain truncation is equivalent to time-domain (in this case space-domain) sampling.

5. FPS-PCG FOR IRREGULAR GEOMETRIES

A limitation in both GFS and FPS is that they can only solve a rectangular 3D domain with layered materials. For GFS, this limitation comes from the need for an analytical form of the Green function like (28). For FPS, this comes from condition (8).

As discussed in Section 1, an accurate thermal model of a die with its spreader and heat sink should be a nonrectangular domain, and that significant errors can be introduced if the spreader and sink are ignored or simplistically modeled as a convective surface of the die.

To solve nonrectangular models, and in general to solve thermal models with material or geometry irregularities, we propose a thermal analysis method called FPS-PCG: FPS-preconditioned Conjugate Gradient.

This method is built upon prior works: the "two-problem approach" from Johnson et al. [1984] Xu et al. [2005], and the "boundary iteration process" from Shi et al. [2009]. The common idea is that if we need to solve $A\mathbf{x} = \mathbf{b}$ where A does not satisfy the conditions of FPS/GFS, we can separate A into two parts. We have

$$A = A' + B, \tag{38}$$

where A' is a system that can be handled efficiently with FPS or GFS, and B represents the abnormities and is much sparser and/or smaller in value than A'.

The proposed FPS-PCG works by using FPS as a preconditioner for Preconditioned Conjugate Gradient (PCG). Given any right-hand-side vector $\mathbf{v}$, FPS can provide an exact solution to $A'\mathbf{x} = \mathbf{v}$ in $O(N \cdot \log N)$ time, which would be an approximate solution to $A\mathbf{x} = \mathbf{v}$. Since any approximate solving process can serve as a preconditioner [Saad 2003], it is natural to put FPS and PCG together.

Figure 3 illustrates FPS-PCG on the pyramid thermal model from Heriz et al. [2007]. Note that FPS-PCG is not limited to solving pyramid-shaped domains, and can handle other nonrectangular models, for example, those that include fins under the sink. The model in Figure 3(a) is expanded to form the rectangular approximation in Figure 3(b).

Like in any PCG scheme, the better approximation FPS provides, the less PCG iterations are needed to converge. It can be argued that FPS on Figure 3(b) provides a good approximation; assuming that the spreader and sink design provides reasonably
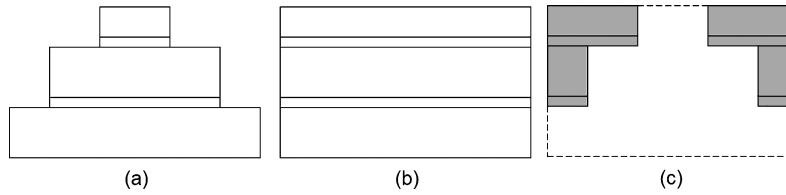
Fig. 3.   Applying FPS-PCG on the pyramid thermal model. (a) Original system *A*. (b) Approximate system *A'*. (c) Abnormity system *B*.

good thermal performance, they should be such that heat flows would remain relatively constant even if the spreader or the die was widened, as is the case in Figure 3(b). The performance of FPS-PCG will be verified in the next section.

Given the relation between FPS and GFS from Section 4, in principle it is also possible to use GFS as a preconditioner and hence GFS-PCG. In practice, this requires a GFS solver that takes a full residual vector as power distribution and outputs a full 3D temperature profile like FPS.

FPS-PCG is also applicable to transient thermal analysis. As discussed in Section 3.2, a transient analysis involves solving a linear system per time step, and, in the case of a nonrectangular domain, FPS-PCG can be used to solve each time step efficiently.

Finally, as is done similarly in Zhan and Sapatnekar [2007], it is possible to solve part of the domain in finer levels of granularity. We obtain, from an FPS-PCG solution, the temperature profile at the thermal contact surface of the die, that is, the interface between the top and middle sections of the pyramid. Then we can solve the die alone and with this temperature profile as the boundary temperatures. Since the die is a rectangular domain, this can be done with one pass of FPS or GFS efficiently, and therefore we can afford to solve with much finer resolution.

6. RESULTS

Two chip testcases are used in this section.

(1) *Testcase no. 1* uses a power map of the POWER6 microprocessor [Jimenez et al. 2010; Le et al. 2007; Stolt et al. 2008] representing one scenario with total power dissipation of 185W. Actual geometries of its die, spreader, and sink are used. Material properties are assumed to be the same as in Heriz et al. [2007]. Figure 4 shows the floorplan of the chip [Le et al. 2007].

(2) *Testcase no. 2* uses an artificially generated power map to model a scenario of a four-core chip, with one core idle, one core with peak load, and two others with median loads; total power is 175W (see Figure 5). The geometries of die, spreader, and sink, as well as material properties, are the same as in Heriz et al. [2007].

Let us first validate, with Testcase no. 1, the need for thermal analysis that incorporates thermal mounts of a chip. Figure 6 demonstrates the difference between thermal analysis with a nonrectangular model and that with a rectangular model. For fairness, the convection coefficient at the thermal contact surface of the die in the rectangular model is adjusted such that Figure 6(a) and Figure 6(b) have the same average temperature. Note that there is significant difference between the shape of temperature distribution between Figure 6(a) and Figure 6(b), and between the two curves in Figure 6(c), and this difference is acute in the hottest regions. This verifies the need for thermal analysis engine, such as the proposed FPS-PCG, for nonrectangular domains.

Then let us validate, again with Testcase no. 1, the accuracy of the FPS-PCG method by checking its solutions against those by a commercial finite element analysis tool ANSYS. Figure 7 plots the maximum error and average error in temperature profiles
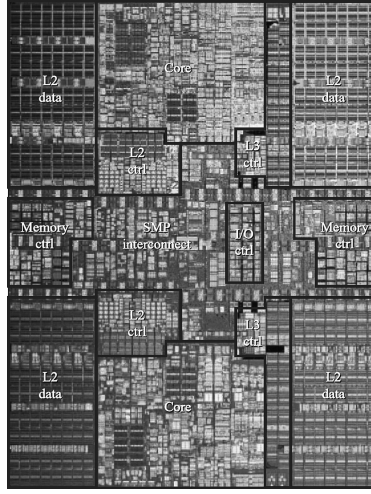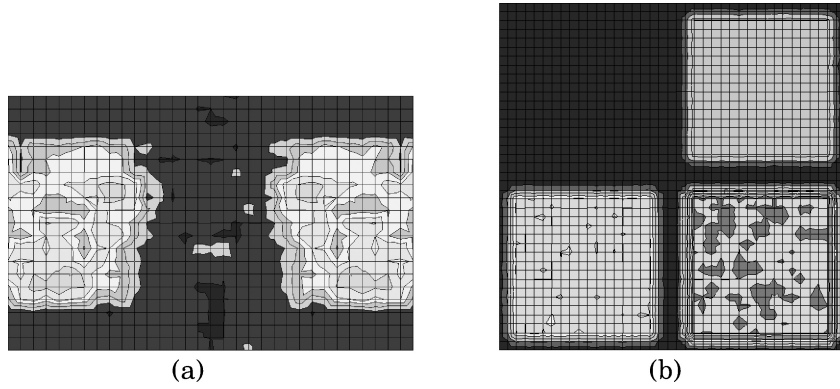
Fig. 4.   POWER6 floorplan [Le et al. 2007].

Fig. 5.   Power maps used for (a) Testcase no. 1, and (b) Testcase no. 2.

computed by FPS-PCG as functions of the granularity of its analysis, measured by the number of finite difference sample points at the device layer. With finer granularity, error quickly comes down to a range that is acceptable in practice; in fact, based on a 30-by-30 solution by FPS-PCG (in 4.93 seconds runtime), we can interpolate a temperature profile that matches the ANSYS-based commercial thermal model with a max difference of 0.93 degree and an average difference of 0.36 degree.

Now let us demonstrate, using both testcases, the efficiency and scalability of the FPS-PCG method. Finite difference discretization is applied on both testcases with different resolutions, resulting in finite difference matrices m1 to m9 in Table I for the Testcase no. 1, and m10–m19 in Table II for the Testcase no. 2. Their dimensions range from 9.83e3 in the coarsest resolution to 7.68e6 in the finest resolution.

Tables I and II compare the performance of FPS-PCG against a direct solver and an Incomplete Cholesky preconditioned Conjugate Gradient (ICCG) solver. The direct solver is based on Cholesky factorization with Approximate Minimum Degree (AMD) ordering [Amestoy et al. 1996]. The ICCG solver is based on incomplete Cholesky factorization with zero fill-in, which is known as IC(0) or the symmetric version of ILU(0) [Saad 2003], and with AMD ordering as well. All three solvers are coded in Matlab, and all runtimes are measured on a 64-bit Linux workstation with 8 CPUs at
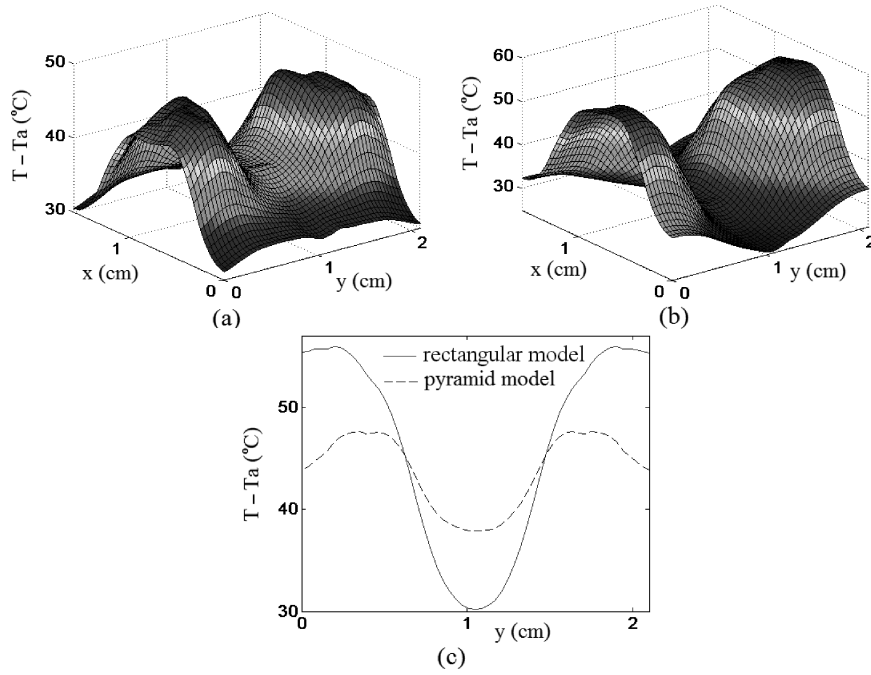
Fig. 6. (a) Chip temperature profile by nonrectangular model. (b) Chip temperature profile by rectangular model. (c) Temperature curves across vertical centerline of the chip by both models. Ta is ambient temperature.

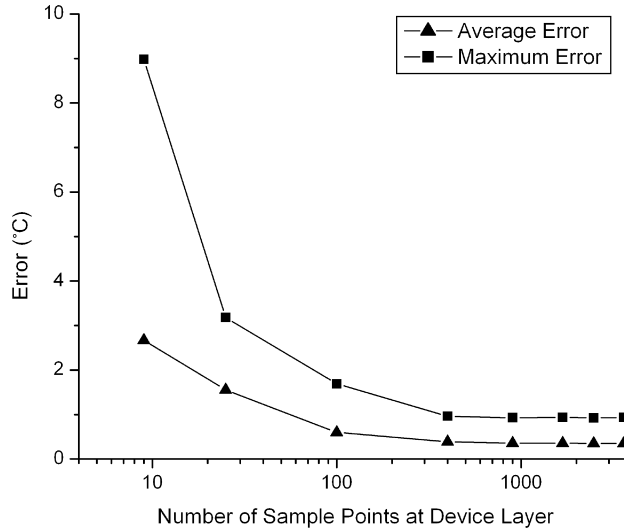Fig. 7. Maximum and average errors of FPS-PCG as functions of the number of finite difference sample points at the device layer.

2.9 GHz frequency and 80GB memory. The direct solver uses 5-7 CPUs during its runs, while ICCG and FPS-PCG both use a single CPU. The ICCG runtimes do not include the overhead of building its IC(0) preconditioner. For both ICCG and FPS-PCG, the convergence criterion is 1e-6 error tolerance.

Table I. Runtime Comparison of FPS-PCG against Direct Solver and
ICCG on Testcase No. 1

Matrix	N	Direct Solver T (sec)	ICCG I	ICCG T (sec)	FPS-PCG I	FPS-PCG T (sec)
m1	9.83e3	0.13	159	0.28	8	0.25
m2	3.93e4	1.85	227	1.14	10	0.73
m3	1.57e5	8.35	343	6.82	10	2.18
m4	3.54e5	38.76	458	22.96	11	4.93
m5	6.60e5	73.10	575	57.14	11	8.38
m6	9.73e5	102.93	571	83.69	11	11.96
m7	1.41e6	212.88	776	172.20	12	18.65
m8	2.42e6	438.14	980	370.66	12	30.81
m9	4.54e6	1013.83	1360	969.13	13	64.78

N is the dimension of a matrix; T is the runtime; I is the number of
conjugate-gradient iterations. The direct solver uses 5-7 CPUs during
its runs, while ICCG and FPS-PCG both use a single CPU.

Table II. Runtime Comparison of FPS-PCG against Direct Solver and
ICCG on Testcase No. 2

Matrix	N	Direct Solver T (sec)	ICCG I	ICCG T (sec)	FPS-PCG I	FPS-PCG T (sec)
m10	1.33e4	0.63	62	0.14	7	0.27
m11	5.33e4	3.73	76	0.58	8	0.77
m12	2.13e5	14.38	114	3.17	10	2.82
m13	4.80e5	62.55	151	12.11	10	6.01
m14	8.53e5	117.41	188	25.87	10	10.19
m15	1.33e6	168.69	221	39.52	10	15.05
m16	1.92e6	222.27	258	81.41	11	23.01
m17	3.41e6	551.71	342	172.99	11	42.52
m18	5.33e6	934.74	425	336.72	11	66.77
m19	7.68e6	N/A	512	594.89	12	102.12

N is the dimension of a matrix; T is the runtime; I is the number of
conjugate-gradient iterations. The direct solver uses 5-7 CPUs during
its runs, while ICCG and FPS-PCG both use a single CPU.

As can be observed in Tables I and II, the direct solver runtime increases at the fastest
rate among the three solvers, and in fact its memory consumption becomes impractical
first and is the reason for its absence for testcase m19. Comparing with ICCG, the FPS-
PCG scales better as the matrix size increases, and its advantage widens, eventually
with approximately 15X speedup over ICCG on m9 (Testcase no. 1), and approximately
6X speedup over ICCG on m19 (Testcase no. 2). This trend is visualized in Figure 8.

A closer look at Tables I and II reveals the reason for FPS-PCG's efficiency and
scalability: it needs consistently less PCG iterations to converge to the given accuracy.
The computational complexity per iteration, which can be measured by T/I in the two
tables, is $O(N)$ for ICCG and $O(N\log N)$ for FPS-PCG. However, as matrix dimension
increases from m1 to m9, the number of iterations needed, I in the two tables, increases
from 159 to 1360 for ICCG, while increase only from 8 to 13 for FPS-PCG. This verifies
the claim in Section 5 that the FPS process is a good approximation to the original linear
system, and therefore serves as an effective and scalable preconditioner for PCG.

Table III studies the relation between the performance of the proposed FPS precon-
ditioner and the degree of irregularity in the system. We take benchmark m4, and
artificially generated benchmarks m4_1 – m4_4 with similar finite difference matrix di-
mensions but with increasing irregularities. Specifically, m4_1 is a rectangular model;
m4_2 uses smaller spreader and sink than m4 and hence is closer to rectangular than
m4; m4_3 uses larger spreader and sink than m4 and hence is more irregular than m4;
m4_4 is m4_3 plus nonuniform thermal conductivities in certain layers to represent the
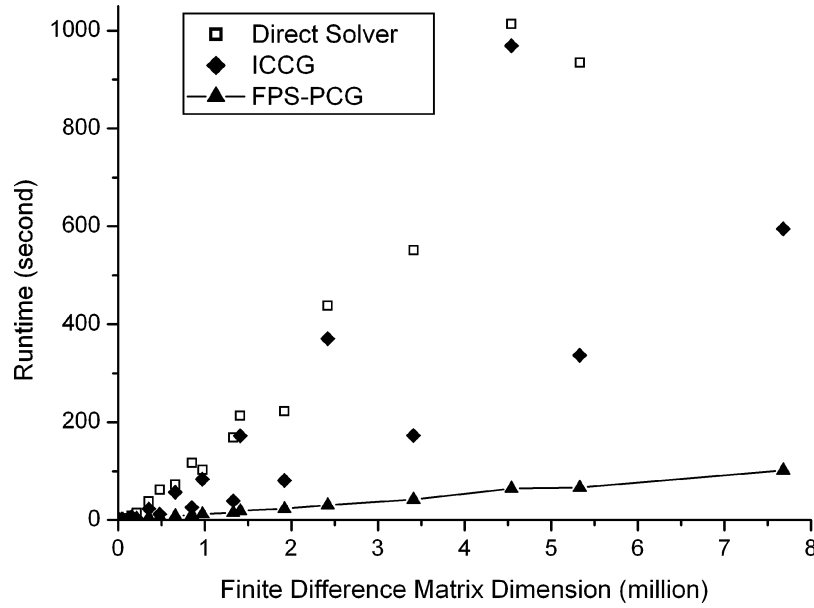
Fig. 8. Runtime T as a function of matrix dimension N for the results in Tables I and II. The direct solver uses 5–7 CPUs during its runs, while ICCG and FPS-PCG both use a single CPU.

Table III. Preconditioning Performance
on Benchmarks with Increasing
Irregularities

Matrix	N	$C1$	$C2$
m4_1	3.56e5	5.73e4	1
m4_2	3.57e5	5.11e4	12.32
m4	3.54e5	4.88e4	15.55
m4_3	3.64e5	4.87e4	16.40
m4_4	3.64e5	4.88e4	20.43

N is the dimension of a matrix; $C1$ is the condition number of the original system; $C2$ is the condition number after preconditioning.

effect of varying metal density in routing layers (or the effect of through-silicon-via distribution in a 3D design). In the extreme case of m4_1, FPS solves the system exactly, hence condition number becomes 1. Table III shows a clear trend that, as the system contains more irregularities and increasingly deviates from a rectangular domain with layered materials, the quality of the FPS preconditioning decreases. It also shows that the degradation is moderate on m4_4, and hence FPS-PCG would likely remain competitive even when solving detailed models that include intralayer material variation.

As discussed in Section 5, after solving a full model, we have the temperature distribution at the thermal contact surface of the die, and therefore can use that as an ambient temperature distribution to solve the die again with a rectangular model and with even finer resolution, which can be done efficiently using the Green function method or the FPS method of Section 3. Figure 9(a) plots a 1200-by-1300 temperature profile, which is computed by a single pass of the FPS method in 7.31 seconds. As expected, given the relation discussed in Section 4 between Green function methods and FPS, this is consistent with the data reported in Zhan and Sapatnekar [2007].
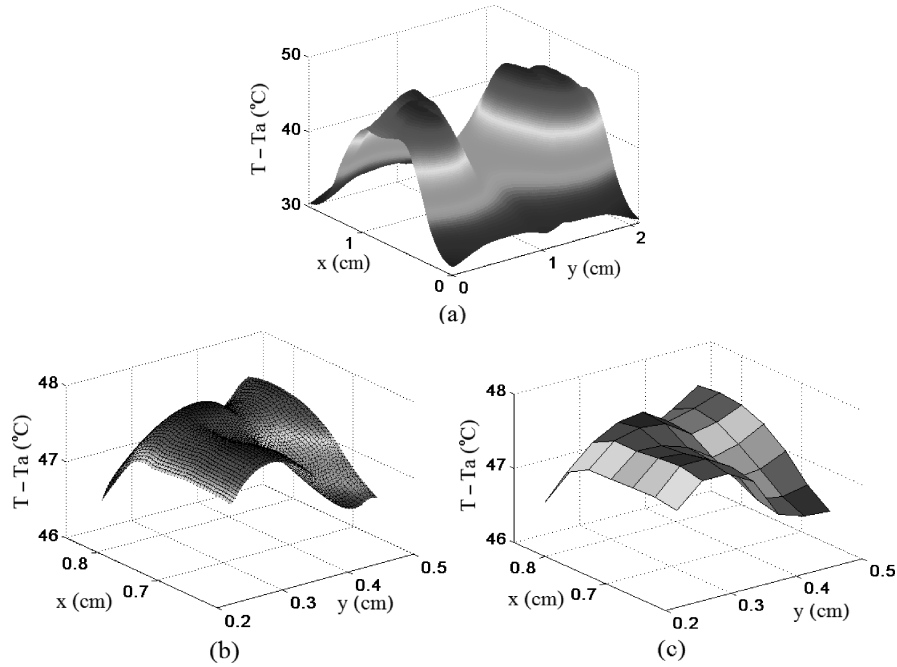
Fig. 9. (a) A 1200-by-1300-resolution chip temperature profile by solving a die-only rectangular model with ambient temperature distribution provided by solving a full model. (b) The left hotspot region in (a). (c) The left hotspot region in Figure 6(a).

Table IV. Runtime of Transient Thermal
Analysis by FPS-PCG

Matrix	N	S	T (sec)
RC1	9.83e3	300	45.14
RC2	3.93e4	300	141.05
RC3	1.57e5	300	479.98
RC4	3.54e5	300	1004.28

N is the dimension of a matrix; S is the
number of time steps; T is the runtime.

However, unlike Zhan and Sapatnekar [2007], the FPS-PCG method can also handle irregular geometries. Comparing Figure 6(a) and 9(a), Figures 9(b), and 9(c), a general agreement is observed, and the high-resolution solution provides more details, which are useful particularly at hotspot regions.

Finally, to verify the proposed solver on transient analysis, we derived four benchmarks RC1 – RC4 based on m1 – m4 respectively and generated artificial time-varying power distributions. We solve them with FPS-PCG for each time step and Table IV shows runtime measurements, where the time step size is 0.1ms. It can be observed that runtime per time step is smaller than Table I; the reason is that the extra diagonal components from thermal capacitances lower the condition numbers of the matrices, and FPS-PCG converges in less iterations.

7. CONCLUSIONS

Fast and accurate thermal analysis is an important step in integrated circuit design. Thermal characteristics not only affect the performance, they also determine the energy efficiency, cost, and reliability of the resulting design. This article studies a class

of eigendecomposition-based fast Poisson solvers for chip-level thermal analysis. The proposed FPS-PCG solver demonstrates superior efficiency and scalability in solving realistic nonrectangular thermal models. This article also proves a strong mathematical relation between fast Poisson solvers and Green-function-based methods. We demonstrate the proposed FPS-based approach on thermal analysis of an industrial microprocessor, showing accurate results verified by a commercial tool, and that it solves a system of dimension 4.54e6 in only 13 conjugate gradient iterations, with a runtime of 65 seconds, a 15X speedup over the popular ICCG solver.

REFERENCES

AMESTOY, P. R., DAVIS, T. A., AND DUFF, I. S. 1996. An approximate minimum degree ordering algorithm. *SIAM J. Matrix Anal. Appl*. 886–905.

ANSYS THERMAL MODELING REFERENCE. 2012. http://www1.ansys.com/customer/content/documentation/121/ans_the.pdf.

BAGNOLI, P. E., CASAROSA, C., AND STEFANI, F. 2007. DJOSER: Analytical thermal simulator for multilayer electronic structures. Theory and numerical implementation. In *Proceedings of the International Conference on Thermal Issues in Emerging Technologies, Theory and Applications*. 111–118.

COSTA, J. P., CHOU, M., AND SILVEIRA, L. M. 1999. Efficient techniques for accurate modeling and simulation of substrate coupling in mixed-signal IC's. *IEEE Trans. Comput. Aided Des. Integr. Circ. Syst. 18*, 5, 597–607.

GHARPUREY, R. AND MEYER, R. G. 1996. Modeling and analysis of substrate coupling in integrated circuits. *IEEE J. Solid State Circ. 31*, 3, 344–353.

HERIZ, V. M., PARK, J., KEMPER, T., KANG, S., AND SHAKOURI, A. 2007. Method of images for the fast calculation of temperature distributions in packaged VLSI chips. In *Proceedings of the 13th International Workshop on Thermal Investigation of ICs and Systems*. 18–25.

HUANG, W., GHOSH, S., VELUSAMY, S., SANKARANARAYANAN, K., SKADRON, K., AND STAN, M. R. 2006. HotSpot: A compact thermal modeling methodology for early-stage VLSI design. *IEEE Trans. VLSI. Syst. 14*, 5, 501–513.

JIMENEZ, V., GIOIOSA, R., KURSUN, E., CAZORLA, F. J., CHER, C., BUYUKTOSUNOGLU, A., BOSE, P., AND VALERO, M. 2010. Trends and techniques for energy efficient architectures. In *Proceedings of the IEEE/IFIP VLSI System on Chip Conference*. 276–279.

JOHNSON, T. A., KNEPPER, R. W., MARCELLO, V., AND WANG, W. 1984. Chip substrate resistance modeling technique for integrated circuit design. *IEEE Trans. Comput. Aided Des. Integr. Circ. Syst. 3*, 126–134.

KARYPIS, G. AND KUMAR, V. 1995. METIS: Unstructured graph partitioning and sparse matrix ordering system, version 2.0. http://glaros.dtc.umn.edu/gkhome/views/metis.

LE, H. Q., STARKE, W. J., FIELDS, J. S., O'CONNELL, F. P., NGUYEN, D. Q., RONCHETTI, B. J., SAUER, W. M., SCHWARZ, E. M., AND VADEN, M. T. 2007. IBM POWER6 microarchitecture. *IBM J. Res. Devel. 51*, 6, 639–662.

LI, P., PILEGGI, L. T., ASHENGHI, M., AND CHANDRA, R. 2004. Efficient full-chip thermal modeling and analysis. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (Digest of Technical Papers)*. 319–326.

MIZUNUMA, H., YANG, C., AND LU, Y. 2009. Thermal modeling for 3D-ICs with integrated microchannel cooling. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (Digest of Technical Papers)*. 256–263.

MIKNEJAD, A. M., GHARPUREY, R., AND MEYER, R. G. 1998. Numerically stable green function for modeling and analysis of substrate coupling in integrated circuits. *IEEE Trans. Comput. Aided Des. Integr. Circ. Syst. 17*, 4, 305–315.

SAAD, Y. 2003. *Iterative Methods for Sparse Linear Systems*. SIAM, Philadelphia, PA.

SABRY, M. M., COSKUN, A. K., ATIENZA, D., ROSING, T. S., AND BRUNSCHWILER, T. 2011. Energy-Efficient multi-objective thermal control for liquid-cooled 3-D stacked architectures. *IEEE Trans. Comput. Aided Des. Integr. Circ. Syst. 30*, 12, 1883–1896.

SHI, J., CAI, Y., HOU, W., MA, L., TAN, S. X. D., HO, P., AND WANG, X. 2009. GPU friendly fast poisson solver for structured power grid network analysis. In *Proceedings of the IEEE/ACM Design Automation Conference (DAC'09)*. 178–183.

SRIDHAR, A., VINCENZI, A., RUGGIERO, M., BRUNSCHWILER, T., AND ATIENZA, D. 2010. 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 (Digest of Technical Papers)*. 463–470.

STOLT, B., MITTLEFEHLDT, Y., DUBEY, S., MITTAL, G., LEE, M., FRIEDRICH, J., AND FLUHR, E. 2008. Design and implementation of the POWER6 microprocessor. *IEEE J. Solid State Circ. 43*, 1, 21–28.

SWARZTRAUBER, P. N. 1977. The methods of cyclic reduction, Fourier analysis and the FACR algorithm for the discrete solution of Poisson's equation on a rectangle. *SIAM Rev. 19,* 3, 490–501.

WANG, B. AND MAZUMDER, P. 2007. Accelerated chip-level thermal analysis using multilayer Green's function. *IEEE Trans. Comput. Aided Des. Integr. Circ. Syst. 26*, 2, 325–344.

XU, C., GHARPUREY, R., FIEZ, T. S., AND MAYARAM, K. 2005. A Green function-based parasitic extraction method for inhomogeneous substrate layers. In *Proceedings of the IEEE/ACM Design Automation Conference (DAC'05)*. 141–146.

ZHAN, Y. AND SAPATNEKAR, S. S. 2007. High efficiency Green function based thermal simulation algorithms. *IEEE Trans. Comput. Aided Des. Integr. Circ. Syst. 26*, 9, 1661–1675.