Fast Poisson Solvers for Thermal Analysis

Haifeng Qian
IBM T. J. Watson Research Center
Yorktown Heights, NY
qianhaifeng@us.ibm.com

Sachin S. Sapatnekar
Department of Electrical and Computer Engineering
University of Minnesota
Minneapolis, MN
sachin@ece.umn.edu

Abstract— **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 non-rectangular 3D domain. This paper 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 $O(N\log N)$ 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 non-rectangular 3D domains efficiently. We demonstrate that this approach solves a system of dimension 5.33e6 in only 11 Conjugate Gradient iterations, with a runtime of 171 seconds, a 6X speedup over the popular ICCG solver.**

Keywords-thermal analysis; fast Poisson solver; Green function

I. Introduction

With technology scaling and on-chip power density increasing, the ability to obtain an accurate full-chip temperature profile under various workload scenarios 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. Thermally-driven optimizations require a large number of thermal solves with different power dissipation distributions, and demand extremely high efficiency from the thermal analysis engine.

Fast chip-level thermal analysis has been the subject of many prior works [5][7][12][14]. Most 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 to be adiabatic [7][14] — this is in the form of the Neumann boundary condition, which specifies temperature gradient (zero for adiabatic surfaces) along the direction that is perpendicular to a surface. The bottom surface of this domain is assumed to be convective, which is the Dirichlet boundary condition specifying temperature values on the bottom surface of the model [7][14], often assumed to be the ambient. A multigrid approach was proposed in [7], which uses finite-difference discretization and a geometric multigrid solver. Several Green-function-based algorithms were proposed in [12][14], which use 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 [3][4][8] on the problems of substrate electrical analysis, which also involves Poisson's equation on a 3D rectangular domain with layered materials.

One limitation of solving a rectangular domain, the die only, is the simplistic assumption about bottom surface, where the temperatures are far from uniform and vary greatly depending on packaging structures underneath. A typical configuration uses a copper heat spreader, a copper heat sink and thermal interface materials layers. It is possible to approximate these thermal mounts with an effective heat transfer coefficient [14], but such an approximation may incur substantial error — this was shown in [5] to be tens of degrees differences and distorted shapes of the temperature profiles. Hence it is important to simulate a die with its thermal mounts, which requires solving Poisson's equation on a non-rectangular 3D domain [2][5].

This work was supported in part by the NSF under award CCF-0634802.

Fig. 1 illustrates the pyramid-shaped model in [5]. This model will be used in Section 4 and Section 5; however, our solver in Section 4 is not limited to solving pyramid-shaped domains, and can handle other non-rectangular models, e.g., those that include fins under the sink.

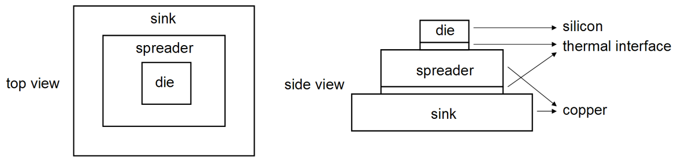

Figure 1. The pyramid thermal model from [5].

The difficulty of a non-rectangular 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. An imaging technique was proposed in [5] as an approximate alternative.

This paper studies eigendecomposition-based fast Poisson solvers for chip-level thermal analysis, and throughout this paper we will use **FPS** as a shorthand notation for these solvers. The idea of FPS has been known for a long time [9][11], but it has found limited usage in practical applications. In the literature, e.g. [9][10], 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 [11].

Section 2 demonstrates a FPS solver for 3D rectangular thermal models with mixed boundary conditions as in [7][14]. The conventional wisdom has been that FPS solvers were an entirely different class of solvers from other known solvers. Section 3 proves, for the first time, a strong relation — in fact, an equivalence under certain conditions — between FPS and Green-function-based methods. Section 4 presents FPS-Preconditioned Conjugate Gradient method that solves non-rectangular domains efficiently.

II. 3D FPS

In this section, we study the use of FPS to solve a 3D finite difference matrix, $A\mathbf{x}=\mathbf{b}$, of a rectangular domain with layered materials and mixed boundary conditions. To do so, we expand the applicability of FPS from the well-known 2D domains with Dirichlet conditions [9]. The following is a relaxed sufficient condition for FPS.

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

such that

$B_{i,j}, \forall i, j$ are square matrices with the same dimension M;

$B_{i,j}, \forall i, j$ have the same eigenvectors.

Let a and b be the dimensions of the 3D 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 inside cell (m,n,l). We assume a natural ordering in $A\mathbf{x}=\mathbf{b}$, in other words:

$$x_{m+n\cdot D_x + l\cdot D_x \cdot D_y} = T_{m,n,l} - T_a \tag{2}$$

$$b_{m+n\cdot D_x + l\cdot D_x \cdot D_y} = P_{m,n,l} \tag{3}$$

where T_a is the ambient temperature at the bottom of the heat sink. As in the thermal models in [7][14], we have Neumann conditions on five surfaces, and Dirichlet boundary condition on the z=0 surface. Fitting matrix A to (1) with $M=D_x$, $K=D_y \cdot D_z$, we have special forms for $B_{i,j}$:

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

$$\text{where } \alpha_i \text{ and } \beta_i \text{ are scalars,} \quad G = \begin{pmatrix} -1 & -1 & 0 & \cdots & 0 & 0 & 0 \\ -1 & 0 & -1 & \cdots & 0 & 0 & 0 \\ 0 & -1 & 0 & \ddots & 0 & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \ddots & 0 & -1 & 0 \\ 0 & 0 & 0 & \cdots & -1 & 0 & -1 \\ 0 & 0 & 0 & \cdots & 0 & -1 & -1 \end{pmatrix} \tag{4}$$

$$\forall i \ne j, \; B_{i,j} = \gamma_{i,j} \cdot I$$

$$\text{where } \gamma_{i,j} \text{ is a scalar.}$$

Note that the above matrices have the same eigenvectors, which are eigenvectors of G; therefore condition (1) is satisfied. Also note that the $B_{i,i}$ formulation in the above equation is different from the well known ones in [9][10]; 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 [9][10]. It can be verified that the eigenvalues and eigenvectors of G are:

$$\forall 1 \le i \le M, \qquad \lambda_i = -2\cos\left(\frac{(i-1)\cdot\pi}{M}\right)$$
$$\mathbf{q}_1 = \sqrt{\frac{1}{M}} \cdot \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix} \qquad \forall i > 1, \; \mathbf{q}_i = \sqrt{\frac{2}{M}} \cdot \begin{pmatrix} \cos\left(\frac{(i-1)\cdot\pi}{2M}\right) \\ \cos\left(\frac{3\cdot(i-1)\cdot\pi}{2M}\right) \\ \vdots \\ \cos\left(\frac{(2M-1)\cdot(i-1)\cdot\pi}{2M}\right) \end{pmatrix} \tag{5}$$

Let matrix Q (which is orthonormal, i.e., $Q^TQ = QQ^T = I$) be:

$$Q = \begin{pmatrix} \mathbf{q}_1 & \mathbf{q}_2 & \cdots & \mathbf{q}_M \end{pmatrix} \tag{6}$$

To solve $A\mathbf{x}=\mathbf{b}$, let us do the following transformation.

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

$$\begin{pmatrix} \Lambda_{1,1} & \Lambda_{1,2} & \cdots & \Lambda_{1,K} \\ \Lambda_{2,1} & \Lambda_{2,2} & \cdots & \Lambda_{2,K} \\ \vdots & \vdots & \ddots & \vdots \\ \Lambda_{K,1} & \Lambda_{K,2} & \cdots & \Lambda_{K,K} \end{pmatrix} \cdot \tilde{\mathbf{x}} = \tilde{\mathbf{b}} \tag{7}$$

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

where the $\Lambda_{i,j}$ matrices take the following diagonal form:

$$\forall i, \qquad \Lambda_{i,i} = Q^T B_{i,i} Q = \alpha_i \cdot I + \beta_i \cdot \begin{pmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_M \end{pmatrix} \tag{8}$$

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

Now that the new left-hand-side matrix in (7) has a block form where each block is a diagonal submatrix, such an overall system 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 (1), each of them can again be transformed and further decoupled into smaller sets of equations.

After $\tilde{\mathbf{x}}$ is obtained, solution $\mathbf{x}$ to the original system is simply:

$$\mathbf{x} = \begin{pmatrix} Q & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & Q \end{pmatrix} \tilde{\mathbf{x}} \tag{9}$$

The advantage of FPS lies in the fact that Q has a special structure such that a fast way exists to evaluate $\tilde{\mathbf{b}}$ from $\mathbf{b}$ by (7), and $\mathbf{x}$ from $\tilde{\mathbf{x}}$ by (9). In fact, for any vector $\mathbf{v}$, the computation of $Q^T\mathbf{v}$ and $Q\mathbf{v}$ can be performed by FFT and with $O(M \cdot \log M)$ complexity.

Let us investigate the complexity of FPS on this thermal analysis. The cost of computing $\tilde{\mathbf{b}}$ from $\mathbf{b}$ by equation (7), 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 equation (9), 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 equation (7) as D_x separate systems, each with dimension $D_y \cdot D_z$.

Studying (4)(7)(8) 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 (1)(4). Therefore, we can apply the FPS procedure on each of them on the y direction (note the 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 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.

III. 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 2 and Green-function-based solvers [4][8][14]. For clarity, we only compare with the full-chip thermal analysis method, Algorithm II, in [14]. The relation to other variations of Green-function-based methods follows similar derivations. For the rest of the paper, 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 [14], that heat sources are located on discrete horizontal planes. Also, like [14], we assume that each plane is divided into $D_x \times D_y$ rectangular grid cells of equal size, and the power density inside each cell is uniform. Therefore, the power density distribution function is in the following piecewise constant form, nonzero only at a finite set of z' values.

$$P(x',y',z') = \sum_{m=0}^{D_x-1}\sum_{n=0}^{D_y-1} P_{m,n}(z') \cdot \Theta\left(x' - (m+0.5)\frac{a}{D_x}, y' - (n+0.5)\frac{b}{D_y}\right) \tag{10}$$

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

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.

A. *FPS formulations*

This section gives detailed FPS formulations based on the principles of Section 2, to be compared with GFS ones in Section 3.2.

Let us use the same formation $A\mathbf{x}=\mathbf{b}$ as (2)(3), and the same x and y discretization as in (10). Let the z coordinates of cell centers be z_0, z_1, …, z_{D_z-1}. For clarity of presentation, we further assume that heat source planes in (10) coincide these z coordinates. In other words:

$$P_{m,n}(z')=0, \qquad P(x',y',z')=0, \qquad \text{for } z'\notin\{z_1,z_2,\cdots z_{D_z-1}\} \tag{11}$$

And we have the following relation between (3) and (10).

$$b_{m+n\cdot D_x+l\cdot D_x\cdot D_y}=P_{m,n,l}=P_{m,n}(z_l)\cdot\frac{a\cdot b}{D_x\cdot D_y} \tag{12}$$

The first step of FPS is to apply the transformation of (7) with $M=D_x$ and $K=D_y\cdot D_z$, and compute the transformed right hand side:

$$\tilde{\mathbf{b}}=\begin{pmatrix}Q_{D_x}^{\mathrm{T}} & 0 & 0\\ 0 & \ddots & 0\\ 0 & 0 & Q_{D_x}^{\mathrm{T}}\end{pmatrix}\mathbf{b} \tag{13}$$

where $Q_{D_x}^{\mathrm{T}}$ denotes the eigenvector matrix from (5)(6) with $M=D_x$.

The new system (7) is then decoupled into D_x independent systems of linear equations. The ith system is, for $0\le i\le D_x$-1:

$$\tilde{A}_i\tilde{\mathbf{x}}_{\mathbf{i}}=\tilde{\mathbf{b}}_{\mathbf{i}} \tag{14}$$

where $\tilde{A}_i$ is a submatrix of the transformed left hand side in (7), $\tilde{\mathbf{x}}_{\mathbf{i}}$ is a part of $\tilde{\mathbf{x}}$, and $\tilde{\mathbf{b}}_{\mathbf{i}}$ is a part of $\tilde{\mathbf{b}}$, such that

$$\tilde{x}_{i,r}=\tilde{x}_{i+D_x\cdot r},\quad \tilde{b}_{i,r}=\tilde{b}_{i+D_x\cdot r},\quad \forall 0\le r\le D_y\cdot D_z-1 \tag{15}$$

Then we apply the transformation of (7) on (14) with $M=D_y$ and $K=D_z$, and compute another transformed right hand side:

$$\tilde{\tilde{\mathbf{b}}}_{\mathbf{i}}=\begin{pmatrix}Q_{D_y}^{\mathrm{T}} & 0 & 0\\ 0 & \ddots & 0\\ 0 & 0 & Q_{D_y}^{\mathrm{T}}\end{pmatrix}\tilde{\mathbf{b}}_{\mathbf{i}} \tag{16}$$

where $Q_{D_y}^{\mathrm{T}}$ denotes the matrix from (5)(6) with $M=D_y$. (14) becomes

$$\begin{pmatrix}Q_{D_y}^{\mathrm{T}} & 0 & 0\\ 0 & \ddots & 0\\ 0 & 0 & Q_{D_y}^{\mathrm{T}}\end{pmatrix}\tilde{A}_i\begin{pmatrix}Q_{D_y} & 0 & 0\\ 0 & \ddots & 0\\ 0 & 0 & Q_{D_y}\end{pmatrix}\tilde{\tilde{\mathbf{x}}}_{\mathbf{i}}=\tilde{\tilde{\mathbf{b}}}_{\mathbf{i}}\quad,\ \text{where}\ \tilde{\tilde{\mathbf{x}}}_{\mathbf{i}}=\begin{pmatrix}Q_{D_y}^{\mathrm{T}} & 0 & 0\\ 0 & \ddots & 0\\ 0 & 0 & Q_{D_y}^{\mathrm{T}}\end{pmatrix}\tilde{\mathbf{x}}_{\mathbf{i}} \tag{17}$$

Vector $\tilde{\tilde{\mathbf{x}}}_{\mathbf{i}}$ can be solved from (17) 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 $\tilde{\tilde{\mathbf{x}}}_{\mathbf{i}}$, we compute

$$\tilde{\mathbf{x}}_{\mathbf{i}}=\begin{pmatrix}Q_{D_y} & 0 & 0\\ 0 & \ddots & 0\\ 0 & 0 & Q_{D_y}\end{pmatrix}\tilde{\tilde{\mathbf{x}}}_{\mathbf{i}} \tag{18}$$

After having $\tilde{\mathbf{x}}_{\mathbf{i}}$ for $0\le i\le D_x$-1, we can now recover $\tilde{\mathbf{x}}$ based on equation (15). Then the solution to the original system is

$$\mathbf{x}=\begin{pmatrix}Q_{D_x} & 0 & 0\\ 0 & \ddots & 0\\ 0 & 0 & Q_{D_x}\end{pmatrix}\tilde{\mathbf{x}} \tag{19}$$

Finally, the temperature values at each cell are mapped from (2). Overall, the FPS procedure can be summarized as follows.

1. Calculate vector $\tilde{\mathbf{b}}$ from $P_{m,n}(z')$ by (12)(13), using FFT.
2. Calculate vectors $\tilde{\tilde{\mathbf{b}}}_{\mathbf{i}}$ from $\tilde{\mathbf{b}}$ by (15)(16), $0\le i\le D_x$-1, using FFT.
3. Calculate vectors $\tilde{\tilde{\mathbf{x}}}_{\mathbf{i}}$ by solving (17) 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 $\tilde{\tilde{\mathbf{x}}}_{\mathbf{i}}$, $0\le i\le D_x$-1, by (15)(18), using FFT.
5. Calculate vector $\mathbf{x}$ from $\tilde{\mathbf{x}}$ by (19), 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 equation (5) with dimension M. By (5)(12)(13)(15)(16), we have:

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

$$\begin{aligned}
\tilde{\tilde{b}}_{i,j+D_y\cdot l}&=\left(\mathbf{q}_{D_y,j+1}\right)^{\mathrm{T}}\cdot\left(\tilde{b}_{i,D_y\cdot l}\ \ \tilde{b}_{i,D_y\cdot l+1}\ \cdots\ \tilde{b}_{i,D_y\cdot l+D_y-1}\right)^{\mathrm{T}}\\
&=\chi_j\cdot\sqrt{\frac{1}{D_y}}\cdot\sum_{n=0}^{D_y-1}\cos\frac{j\pi(n+0.5)}{D_y}\cdot\tilde{b}_{i,D_y\cdot l+n}\\
&=\chi_j\cdot\sqrt{\frac{1}{D_y}}\cdot\sum_{n=0}^{D_y-1}\cos\frac{j\pi(n+0.5)}{D_y}\cdot\tilde{b}_{i+D_x\cdot n+D_x\cdot D_y\cdot l}\\
&=\chi_j\cdot\sqrt{\frac{1}{D_y}}\cdot\sum_{n=0}^{D_y-1}\cos\frac{j\pi(n+0.5)}{D_y}\cdot\left(\mathbf{q}_{D_x,i+1}\right)^{\mathrm{T}}\cdot\begin{pmatrix}b_{D_x\cdot n+D_xD_y\cdot l}\\ b_{D_x\cdot n+D_xD_y\cdot l+1}\\ \vdots\\ b_{D_x\cdot n+D_xD_y\cdot l+D_x-1}\end{pmatrix}\\
&=\chi_i\cdot\chi_j\cdot\sqrt{\frac{1}{D_xD_y}}\cdot\sum_{m=0}^{D_x-1}\sum_{n=0}^{D_y-1}P_{m,n,l}\cdot\cos\frac{i\pi(m+0.5)}{D_x}\cdot\cos\frac{j\pi(n+0.5)}{D_y}\\
&=\frac{\chi_i\cdot\chi_j\cdot a\cdot b}{D_x^{1.5}\cdot D_y^{1.5}}\cdot\sum_{m=0}^{D_x-1}\sum_{n=0}^{D_y-1}P_{m,n}(z_l)\cdot\cos\frac{i\pi(m+0.5)}{D_x}\cdot\cos\frac{j\pi(n+0.5)}{D_y}
\end{aligned} \tag{20}$$

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 (2)(5)(15)(18)(19), the average temperature of cell (m,n,l) is:

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

$$\begin{aligned}
T_{m,n,l}&=T_a+x_{m+n\cdot D_x+l\cdot D_x\cdot D_y}\\
&=T_a+\sqrt{\frac{1}{D_x}}\cdot\sum_{i=0}^{D_x-1}\chi_i\cdot\cos\frac{i\pi(m+0.5)}{D_x}\cdot\tilde{x}_{i+n\cdot D_x+l\cdot D_x\cdot D_y}\\
&=T_a+\sqrt{\frac{1}{D_x}}\cdot\sum_{i=0}^{D_x-1}\chi_i\cdot\cos\frac{i\pi(m+0.5)}{D_x}\cdot\tilde{x}_{i,n+l\cdot D_y}\\
&=T_a+\sqrt{\frac{1}{D_xD_y}}\cdot\sum_{i=0}^{D_x-1}\sum_{j=0}^{D_y-1}\chi_i\cdot\chi_j\cdot\tilde{\tilde{x}}_{i,j+l\cdot D_y}\cdot\cos\frac{i\pi(m+0.5)}{D_x}\cdot\cos\frac{j\pi(n+0.5)}{D_y}
\end{aligned} \tag{21}$$

where χ_j is as in (20).

B. *GFS formulations*

This section is largely based on Algorithm II in [14]. The GFS approach evaluates the following temperature calculation formula:

$$T(x,y,z)=T_a+\sum_{z'}\int_0^a dx'\int_0^b dy' G(x,y,z,x',y',z')P(x',y',z') \tag{22}$$

where $G(.)$ is the Green function. In the case of Neumann conditions on 4 side surfaces (x=0, y=0, x=a, y=b), by [8][14], it is:

$$G(x,y,z,x',y',z')=\sum_{i=0}^{\infty}\sum_{j=0}^{\infty}C_{i,j}(z,z')\cdot\cos\frac{i\pi x}{a}\cdot\cos\frac{j\pi y}{b}\cdot\cos\frac{i\pi x'}{a}\cdot\cos\frac{j\pi y'}{b} \tag{23}$$

and [8] provided ways to calculate function $C_{i,j}(.)$ analytically.

Suppose we write (10) in the following frequency-domain form.

$$P(x',y',z')=\sum_{i=0}^{\infty}\sum_{j=0}^{\infty}p_{i,j}(z')\cdot\cos\frac{i\pi x'}{a}\cdot\cos\frac{j\pi y'}{b} \tag{24}$$

By (10) and (24), we can derive

$$p_{i,j}(z')=a\cdot b\cdot\mu_{i,j}\cdot\sum_{m=0}^{D_x-1}\sum_{n=0}^{D_y-1}P_{m,n}(z')\cdot\cos\frac{i\pi(m+0.5)}{D_x}\cdot\cos\frac{j\pi(n+0.5)}{D_y} \tag{25}$$

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

Substituting (23) and (24) into (22), we get an alternative to (22):

$$\begin{aligned}
T(x,y,z)&=T_a+a\cdot b\cdot\sum_{z'}\sum_{i=0}^{\infty}\sum_{j=0}^{\infty}\omega_{i,j}\cdot p_{i,j}(z')\cdot C_{i,j}(z,z')\cdot\cos\frac{i\pi x}{a}\cdot\cos\frac{j\pi y}{b}\\
&=T_a+a\cdot b\cdot\sum_{i=0}^{\infty}\sum_{j=0}^{\infty}Z_{i,j}(z)\cdot\cos\frac{i\pi x}{a}\cdot\cos\frac{j\pi y}{b}
\end{aligned} \tag{26}$$

$$\text{where}\quad \omega_{i,j}=\begin{cases}1, & i=j=0\\ 0.5, & i=0,j\neq 0\\ 0.5, & i\neq 0,j=0\\ 0.25, & i\neq 0,j\neq 0\end{cases}\quad,\quad Z_{i,j}(z)=\sum_{z'}\omega_{i,j}\cdot p_{i,j}(z')\cdot C_{i,j}(z,z')$$

Practically, we need to truncate the infinite summations in (26). By truncating the two infinite sums at D'_x and D'_y respectively:

$$T(x,y,z)=T_a+a\cdot b\cdot\sum_{i=0}^{D'_x-1}\sum_{j=0}^{D'_y-1}Z_{i,j}(z)\cdot\cos\frac{i\pi x}{a}\cdot\cos\frac{j\pi y}{b} \qquad (27)$$

In practice, as is done in [14], often we are interested in not the direct outcome of (27), but, by dividing the x-y plane into evenly D''_x by D''_y rectangular cells, average temperatures in each cell. They are:

$$\text{for } 0\le m\le D''_x-1,\ 0\le n\le D''_y-1$$

$$T_{m,n}(z)=\frac{D''_xD''_y}{ab}\int_{x=\frac{am}{D''_x}}^{\frac{a(m+1)}{D''_x}}dx\int_{y=\frac{bn}{D''_y}}^{\frac{b(n+1)}{D''_y}}dyT(x,y,z) \qquad (28)$$

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

$$\text{where }\theta_{i,j}=\begin{cases}\dfrac{1}{D''_xD''_y}, & i=j=0\\ \dfrac{2}{j\pi D''_x}\sin\dfrac{j\pi}{2D''_y}, & i=0,\ j\ne 0\\ \dfrac{2}{i\pi D''_y}\sin\dfrac{i\pi}{2D''_x}, & i\ne 0,\ j=0\\ \dfrac{4}{ij\pi^2}\sin\dfrac{i\pi}{2D''_x}\sin\dfrac{j\pi}{2D''_y}, & i\ne 0,\ j\ne 0\end{cases}$$

Overall, the GFS procedure can be summarized as follows.

1. Calculate function $C_{i,j}(z,z')$ for $0\le i\le D'_x-1, 0\le j\le D'_y-1$, analytically as in [8].
2. Calculate function $p_{i,j}(z')$ by (25), for $0\le i\le D'_x-1$, $0\le j\le D'_y-1$, using FFT.
3. Calculate function $Z_{i,j}(z)$ by (26), for $0\le i\le D'_x-1$, $0\le j\le D'_y-1$.
4. Calculate function $T_{m,n}(z)$ by (28), for $0\le m\le D''_x-1$, $0\le n\le D''_y-1$, using FFT.

C. *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: (D_x, D_y) of the heat source distribution, (D'_x,D'_y) of frequency domain truncation, and (D''_x,D''_y) of temperature output. They are not completely independent choices, for example, it was shown in [14] 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 equation (20), which is steps 1 and 2 of FPS, and equation (25), which is step 2 of GFS. Specifically, both equations represent the same computation and produce values merely differing by a scaling factor:

$$\text{for } 0\le i\le D_x-1,\ 0\le j\le D_y-1,\ 0\le l\le D_z-1 \qquad (29)$$

$$p_{i,j}(z_l)\equiv\frac{\mu_{i,j}\cdot D_x^{1.5}\cdot D_y^{1.5}}{\chi_i\cdot\chi_j}\cdot\tilde{\tilde{b}}_{i,j+D_y\cdot l}$$

Similarly, there is a correspondence between (21), which is steps 4 and 5 of FPS, and (28), which is step 4 of GFS. Both equations represent the same computation and produce the same temperature outcome (subject to numerical errors), based on input values $\tilde{\tilde{\mathbf{x}}}_{\mathrm{i}}$ and $Z_{i,j}(z)$ respectively, which merely differ by a scaling factor:

$$\text{for } 0\le i\le D_x-1,\ 0\le j\le D_y-1,\ 0\le l\le D_z-1 \qquad (30)$$

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

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

The above is not an exact equality, for $\tilde{\tilde{\mathbf{x}}}_{\mathrm{i}}$ and $Z_{i,j}(z)$ are obtained differently: step 3 of FPS computes $\tilde{\tilde{\mathbf{x}}}_{\mathrm{i}}$ 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. However, step 3 of FPS and steps 1 and 3 of GFS do the same task, — they take the same input, as shown in (29), and produce approximately the same output, as shown in (30).

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, 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.

IV. 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 comes from the need for an analytical Green function like (23). For FPS, this comes from condition (1). Recall that an accurate thermal model of a die with its spreader and heat sink should be a non-rectangular domain, and that significant errors can be introduced if the spreader and sink are ignored or simplistically modeled as a convective surface.

To solve non-rectangular models, and in general 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 [6][13], and the "boundary iteration process" from [10]. The common idea is that, to solve $A\mathbf{x}=\mathbf{b}$ where A does not satisfy the conditions of FPS/GFS, we can separate A into two parts:

$$A=A'+B \qquad (31)$$

where A' can be handled efficiently with FPS/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 **v**, FPS can provide in 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 [9], it is natural to put FPS and PCG together.

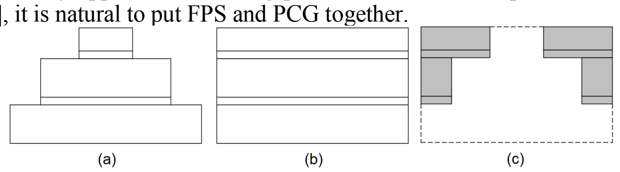

Figure 2. Applying FPS-PCG on the pyramid thermal model. (a) Original system *A*. (b) Approximate system *A'*. (c) Abnormity system *B*.

Fig. 2 illustrates FPS-PCG on the pyramid thermal model from [5]. Note that FPS-PCG is not limited to solving pyramid-shaped domains, and can handle other non-rectangular models, e.g., those that include fins under the sink. The model in Fig. 2(a) is expanded to form the rectangular approximation in Fig. 2(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 provides a good approximation, — assuming that the spreader and sink design provides reasonably good thermal performance, they should be such that heat flows would remain relatively constant even if the spreader or the die was widened, as in Fig. 2(b). The performance of FPS-PCG will be verified in the next section.

Finally, as is done similarly in [14], it is possible to solve part of the domain in finer levels of granularity. We obtain, from a FPS-PCG solution, the temperature profile at the bottom of the die, i.e., 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.

V. RESULTS

For testcases, we use the same geometries of die, spreader and sink, as well as material properties, as in [5]. Nine testcases are generated by discretization of this structure with different resolutions. Consequently, the nine finite difference matrices, m1-m9 in Table 1, have dimensions ranging from 1.33e4 in the coarsest resolution to 5.33e6 in the finest. A heat map is used to model one operation scenario of a four-core chip, with total power 175W.

TABLE I. RUNTIME COMPARISON OF FPS-PCG AGAINST DIRECT SOLVER AND ICCG. N IS THE DIMENSION; T IS THE RUNTIME; I IS THE NUMBER OF CONJUGATE-GRADIENT ITERATIONS.

Matrix	N	**Direct Solver**	**ICCG**		**FPS-PCG**	
		T **(sec)**	I	T **(sec)**	I	T **(sec)**
m1	1.33e4	0.70	62	0.34	7	0.47
m2	5.33e4	5.54	76	2.52	8	1.57
m3	2.13e5	46.24	114	10.56	9	5.65
m4	4.80e5	164.44	151	28.17	10	13.31
m5	8.53e5	388.98	187	62.09	10	22.37
m6	1.33e6	N/A	221	132.48	10	34.84
m7	1.92e6	N/A	258	189.06	11	53.13
m8	3.41e6	N/A	342	439.75	11	103.34
m9	5.33e6	N/A	425	1007.73	11	171.11

Table 1 compares FPS-PCG against a direct solver and an ICCG solver. The direct solver uses approximate minimum degree (AMD) ordering [1]. The ICCG solver is based on incomplete Cholesky factorization with zero fill-in [9], and with AMD ordering. All three solvers are coded in Matlab, and all runtimes are measured on a Linux workstation with 2.4GHz CPU frequency and 6GB memory. The ICCG runtimes do not include preconditioning. For both ICCG and FPS-PCG, the convergence criterion is 1e-6 error tolerance.

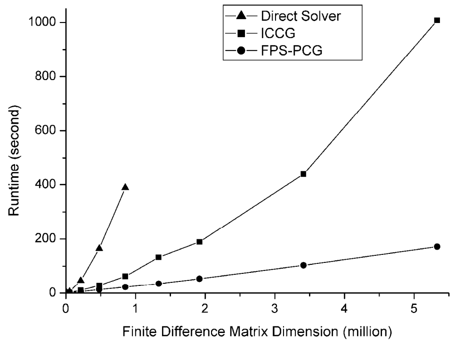

Figure 3. Runtime T as a function of matrix dimension N in Table 1.

As can be observed in Table 1, the direct solver runtime increases at approximately $O(N^{1.5})$ rate for these matrices, and in fact its memory consumption becomes impractical for the last four testcases. FPS-PCG scales better than ICCG as the matrix size increases, and its advantage widens, eventually with approximately 6X speedup on m9.

A closer look at Table 1 reveals the reason for FPS-PCG's efficiency and scalability ─ it needs consistently less PCG iterations to converge. The complexity per iteration, which can be measured by T/I in Table 1, is $O(N)$ for ICCG and $O(N\log N)$ for FPS-PCG. However, as dimension increases from m1 to m9, the number of iterations, I in Table 1, increases from 62 to 425 for ICCG, while increase only from 7 to 11 for FPS-PCG. This verifies the claim in Section 4 that the FPS process is a good approximation to the original linear system, and therefore serves as an effective and scalable preconditioner for PCG.

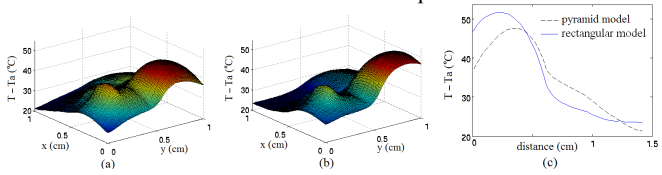

Figure 4. (a) Chip temperature profile by non-rectangular model. (b) Chip temperature profile by rectangular model. (c) Temperature curves across chip diagonal by both models. T_a is ambient temperature at bottom of heat sink.

Fig. 4 demonstrates the difference between rectangular and non-rectangular models. For fairness, convection coefficient at the bottom of the rectangular model is adjusted such that Fig. 4(a) and 4(b) have the same average temperature. Note that there is significant difference between the shape of distribution between Fig. 4(a) and 4(b), and between the two curves in Fig. 4(c), and this difference is especially acute in the hottest regions. This verifies the need for thermal analysis engine, such as FPS-PCG, for non-rectangular domains.

Finally, as discussed in Section 4, after solving a full model, we have the temperature distribution at the bottom of the die, and 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 GFS or FPS of Section 2. Fig. 6(a) plots a 1200 by 1200 temperature profile, which is computed by the FPS method in 17.89 seconds. As expected, given the relation between GFS and FPS, this runtime is consistent with those reported in [14].

Comparing Fig. 4(a) and Fig. 5(a), Fig. 5(b) and Fig. 5(c), a general agreement is observed, and the high-resolution solution provides more details, which are useful at hot-spot regions.

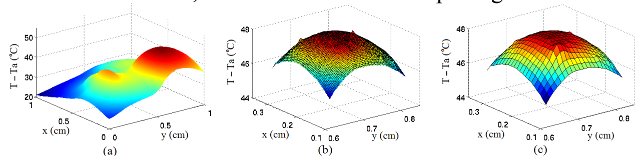

Figure 5. (a) A 1200-by-1200 resolution temperature profile by solving a die-only rectangular model with ambient distribution provided by solving a full model. (b) The hot-spot region in (a). (c) The hot-spot region in Fig. 4(a).

REFERENCES

[1] P. Amestoy, T. A. Davis, and I. Duff, "An approximate minimum degree ordering algorithm," *SIAM J. Matrix Anal. Appl.*, pp. 886-905, 1996.

[2] P. E. Bagnoli, C. Casarosa, and F. Stefani, "DJOSER: Analytical thermal simulator for multilayer electronic structures. Theory and numerical implementation," *Proc. Int'l Conf. Thermal Issues in Emerging Technologies, Theory and Applications*, pp. 111-118, 2007.

[3] J. Costa, M. Chou, and L. M. Silveira, "Efficient techniques for accurate modeling and simulation of substrate coupling in mixed-signal IC's," *IEEE Trans. Computer-Aided Design*, vol. 18, no. 5, pp. 597-607, 1999.

[4] R. Gharpurey and R. G. Meyer, "Modeling and analysis of substrate coupling in integrated circuits," *IEEE Journal of Solid-State Circuits*, vol. 31, no. 3, pp. 344-353, 1996.

[5] V. M. Heriz, J. Park, T. Kemper, S. Kang, and A. Shakouri, "Method of images for the fast calculation of temperature distributions in packaged VLSI chips," *Proc. 13th Int'l Workshop on Thermal Investigation of ICs and Systems*, pp. 18-25, 2007.

[6] T. A. Johnson, R. W. Knepper, V. Marcello, and W. Wang, "Chip substrate resistance modeling technique for integrated circuit design," *IEEE Trans. Computer-Aided Design,* vol. 3, pp. 126-134, 1984.

[7] P. Li, L. T. Pileggi, M. Asheghi, and R. Chandra, "Efficient full-chip thermal modeling and analysis," *Proc. ICCAD*, pp. 319-326, 2004.

[8] A. M. Niknejad, R. Gharpurey, and R. G. Meyer, "Numerically stable Green function for modeling and analysis of substrate coupling in integrated circuits," *IEEE Trans. Computer-Aided Design*, vol. 17, no. 4, pp. 305-315, 1998.

[9] Y. Saad. *Iterative Methods for Sparse Linear Systems*, SIAM, Philadelphia, PA, 2003.

[10] J. Shi, Y. Cai, W. Hou, L. Ma, S. X.-D. Tan, P. Ho, and X. Wang, "GPU friendly fast Poisson solver for structured power grid network analysis," *Proc. ACM/IEEE Design Automation Conference*, pp. 178-183, 2009.

[11] P. N. Swarztrauber, "The methods of cyclic reduction, Fourier analysis and the FACR algorithm for the discrete solution of Poisson's equation on a rectangle," *SIAM Review*, vol. 19, no. 3, pp. 490-501, 1977.

[12] B. Wang and P. Mazumder, "Accelerated chip-level thermal analysis using multilayer Green's function," *IEEE Trans. Computer-Aided Design*, vol. 26, no. 2, pp. 325-344, 2007.

[13] C. Xu, R. Gharpurey, T. S. Fiez, and K. Mayaram, "A Green function-based parasitic extraction method for inhomogeneous substrate layers," *Proc. ACM/IEEE Design Automation Conference*, pp. 141-146, 2005.

[14] Y. Zhan and S. S. Sapatnekar, "High efficiency Green function-based thermal simulation algorithms," *IEEE Trans. Computer-Aided Design*, vol. 26, no. 9, pp. 1661-1675, 2007.