Full-Chip Thermal Analysis for the Early Design Stage via Generalized Integral Transforms

Pei-Yu Huang, Chih-Kang Lin, and Yu-Min Lee, *Member*, *IEEE*
Department of Communication Engineering, National Chiao Tung University Hsinchu 300, Taiwan
pey.cm93g@nctu.edu.tw, is84013@cis.nctu.edu.tw and yumin@cm.nctu.edu.tw

Abstract— **The capability of predicting the temperature profile is critically important for circuit timing estimation, leakage reduction, power estimation, hotspot avoidance, and reliability concerns during modern IC designs.**

This paper presents an accurate and fast analytical full-chip thermal simulator for the early-stage temperature-aware chip design. By using the technique of generalized integral transforms (GIT), our proposed method can accurately estimate the temperature distribution of full-chip with very small truncation points of bases in the spatial domain. We also develop a fast Fourier transform (FFT) like evaluating algorithm to efficiently evaluate the temperature distribution. Experimental results confirm that our GIT based analyzer can achieve an order of magnitude speedup compared with a highly efficient Green's function based method.

I. Introduction

The power density of VLSI circuits increases monotonously as the CMOS technology continuously scales down. Because the power dissipated by circuits converts into heat, as a result, it raises the temperature of dies and induces hot spots. These thermal-related phenomena significantly degrade the performance and reliability of circuits [1]–[6], To precisely predict thermal impacts on design performance, an efficient and accurate thermal analyzer is necessary in the temperature-aware design flow because it is usually a part of simulation kernel in the optimization loop and need to be executed numerous times.

Essentially, existing thermal simulators can be categorized into two classes, numerical and analytical methods. The numerical methods apply the finite difference method (FDM) or finite element method (FEM) to transfer heat equations to RC network equations. Based on the RC network equations, several methods are proposed to improve the run time. For example, the alternating-direction-implicit based method [1], the model order reduction based method [2], and the multi-grid method [3]. Because of the flexibility for dealing with complicated structure, the numerical framework is suitable for the back-end stage of design flow such as the post layout thermal verification.

As pointed out in [4], the temperature-aware design should be brought to the early design stage such as thermal-aware floorplanning and placement. To give a reasonably accurate temperature prediction with little computational effort, they proposed an accurate compact thermal model for modeling equivalent heat transfer coefficients of the pre-layout package and interconnect layers for the boundary conditions of die, and provided a numerical method for the temperature calculating of die.

Although numerical methods can be directly applied to simulate the temperature distribution of the model proposed in [4], they are not suitable for the early temperature-aware design stage because they require the volume meshing of entire substrate even if the devices are usually built within a thin layer close to the top surface of die, and the material of substrate can be treated as homogeneous during the early design stage [5]. Because of the volume meshing, a huge set of linear equations for the uninterested temperature in substrate still need to be handled even if only the temperature distribution close to the device layer is of interest.

On the contrary, analytical methods are good candidates for the early design stage because they avoid directly performing the volume meshing of entire substrate and have closed-form representations for the temperature distribution. One analytical category of thermal solvers is the Green's function based method [6]. In [6], the authors applied the Green's function to the time-independent Possion's equation and used fast Fourier transform (FFT) to evaluate the steady-state temperature distribution. Hence, the computational cost can be only $O(MN\log_2 MN)$, where M, and N are numbers of divisions in the power density map along x-, and y-directions, respectively. However, the convergent rate of their formulation is not fast enough because the generated cosine series based on time-independent Possion's equation [6] can not fully capture the transient characteristics of original heat equations. As shown in [6], the truncation points need to be large enough to achieve small relative error. Furthermore, it is only for steady state temperature calculation, but the transient analysis is also necessary while performing the dynamic thermal management and run-time thermal analysis [4]–[6].

To overcome these shortcomings, our major contributions are

- We improve the convergence rate of analytical solution for steady state temperature distribution and provide a transient temperature simulation by utilizing generalized integral transforms (GIT) [7] to construct a set of spatial bases and their corresponding time-varying coefficients. The proposed method can accurately estimate the temperature distribution of full-chip with very small truncation points (N_x and N_y) of bases in the spatial domain. The experimental results presented in section IV show that N_xN_y can be far less than MN without losing any accuracy compared with [6].
- We develop a FFT like evaluating algorithm to efficiently evaluate the temperature map of all grid cells, and its computational cost is in order of $O(MN\log_2 N_xN_y)$, where N_x, and N_y are truncation points of bases along x-, and y-directions, respectively.

The rest of this paper is organized as follows. First, the thermal modeling for the early design stage is introduced in section II. Then, the full-chip thermal simulation by using the GIT technique, and the proposed temperature evaluating algorithm are described in section III. Finally, the experimental results and conclusions are given in section IV and V, respectively.

II. Thermal modeling for the early design stage

The thermal model for the early design stage can be modeled as a compact structure which consists of the primary heat flow path, the secondary heat flow path, and the heat transfer characteristic of each macro/block on silicon die [4] as shown in Fig. 1. The primary heat flow path is composed of thermal interface material, heat spreader, and heat sink. The secondary heat flow path contains interconnect layers, I/O pads, and print circuit board

This work was supported in part by National Science Council of Taiwan, in part by MoE ATU Program, and in part by SoC Research Center of NCTU. We are grateful to the National Center for High-performance Computing for computer time and facilities.

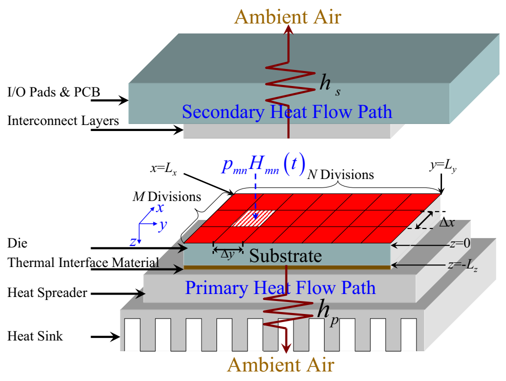

Fig. 1. Compact thermal model of the early design stage.

(PCB). The functional blocks on the die are modeled as many power sources attached on the top surface of die.

The rising temperature $T(\mathbf{r}, t)$ of die corresponding to the ambient temperature can be governed by the following heat transfer equations [5].

$$\nabla \cdot (\kappa(\mathbf{r}) \nabla T(\mathbf{r}, t)) = \sigma(\mathbf{r}) \frac{\partial T(\mathbf{r}, t)}{\partial t}; \mathbf{r} \in D \tag{1}$$

$$\kappa(\mathbf{r}) \frac{\partial T(\mathbf{r}, t)}{\partial n_{b_s}} + h_{b_s} T(\mathbf{r}, t) = f_{b_s}(\mathbf{r}) \tag{2}$$

where $\mathbf{r} = (x, y, z)$, $\kappa(\mathbf{r})$ is the thermal conductivity (W/m·°C) of die, $\sigma(\mathbf{r})$ is the product of the density of matrial and specific heat (J/m^3·°C) of die, ∇ is the diverge operator, D=(0,L_x)×(0,L_y)×($-L_z$,0) is the dimension of die, L_x and L_y are the lateral sizes of die, L_z is the thickness of die, h_{b_s} is the heat-transfer coefficient on the boundary surface, b_s, of die, $f_{b_s}(\mathbf{r})$ is the heat flux function on the boundary surface, and $\partial/\partial n_{b_s}$ is the differentiation along the outward direction normal to the boundary surface.

To provide reasonable accuracy with little computational effort during the early-stage temperature-aware optimization procedure, heat-transfer coefficients on the boundary surfaces of die should be appropriately modeled. Based on the model proposed in [4], the heat transfer coefficients of primary path can be equalized to an effective heat transfer coefficient h_p by combining the effect of each component on the primary path. Since the detail layout of interconnects is not available in the early design stage, [4] modeled the interconnect layer as an equivalent thermal resistance by estimating the density based on the regularity structure assumption of metal and dielectric material. With the model of each interconnect layer, the heat transfer coefficients of secondary path can be simplified to be an equivalent heat transfer coefficient h_s by stacking the thermal resistance of each layer with I/O pads and PCB.

Because of the chip and package structures, the area of vertical surface is strictly less than the area of horizontal surface and the thermal conductivity of air is much less than the values of primary and secondary heat transfer paths. Therefore, the boundary condition of vertical surface is set to be adiabatic [6]. The heat transfer characteristic of functional blocks on the die is modeled as an equivalent heat equation with many power generating sources attached on the top surface of die and substrate.

Finally, although in general, the thermal parameters, $\kappa(\mathbf{r})$ and $\sigma(\mathbf{r})$, of die are position-dependent, the variations of these thermal parameters are usually not significant, and as suggested in [4]–[6], these parameters can be treated as constants while performing the temperature-aware floor-planning and placement.

Based on the above model, the heat diffusion equations of die for the early design stage can be re-written as

$$\kappa \nabla^2 T(x, y, z, t) = \sigma \frac{\partial T(x, y, z, t)}{\partial t}; (x, y, z) \in D \tag{3}$$

$$\left.\frac{\partial T(x, y, z, t)}{\partial x}\right|_{x=0, L_x} = \left.\frac{\partial T(x, y, z, t)}{\partial y}\right|_{y=0, L_y} = 0 \tag{4}$$

$$\kappa \left.\frac{\partial T(x, y, z, t)}{\partial z}\right|_{z=-L_z} = h_p T(x, y, -L_z, t) \tag{5}$$

$$\kappa \left.\frac{\partial T(x, y, z, t)}{\partial z}\right|_{z=0} = h_s T(x, y, 0, t) + p(x, y, t) \tag{6}$$

Here, $p(x, y, t)$ is the power density (W/m^2) on the top surface of die and $T(x, y, z, 0)$ = 0.

By discretizing the power source plane on the top of die into MN grid cells, where M and N are numbers of divisons along x- and y-directions, respectively, the power density $p(x, y, t)$ can be rewritten as

$$p(x, y, t) = \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} p_{mn} \Pi_{mn}(x, y) H_{mn}(t), \tag{7}$$

where $\Pi_{mn}(x, y)$ is the indicate function with nonzero value equaling to 1 only when (x, y) is in $[m\Delta x, (m+1)\Delta x] \times [n\Delta y, (n+1)\Delta y]$, Δx=L_x/M, Δy=L_y/N, m and n are indices of divisions, and p_{mn} and $H_{mn}(t)$ are the average power density and the turning on/off function of grid cell (m, n), respectively.

As calculating the steady state temperature, $H_{mn}(t)$ is a unit step function. Otherwise, it is an instruction specified time interval function [3]. With above government equations, our goal is to get the rising temperature distribution of die corresponding to the ambient temperature.

III. Full-Chip Thermal Simulation

The computational procedure of GIT includes two steps [7]. In the beginning, a set of appropriate bases is generated by a system-compatible auxiliary problem. Several guidelines need to be followed for choosing this auxiliary problem. Firstly, the auxiliary problem should be as similar as the original problem. Secondly, the generated bases have to be completely ortho-normalized to ensure the convergence in mean of the approximated temperature distribution [7]. Finally, the ortho-normal bases should be time independent for efficiency consideration. After bases being constructed, the temperature distribution can be expressed by those bases with suitable time-varying coefficients.

In next two subsections, how to apply the above procedure to the full-chip thermal analysis will be described in detail. After that, the compact formula will be given to calculate the average steady-state temperature distribution, and its convergence rate improvement over the Green's function based method [6] will be pointed out. Finally, we will develop fast evaluating algorithms for our GIT based formulation, and utilize our method to perform the transient thermal simulation.

A. Auxiliary Problem for Generating Appropriate Spatial Bases

The auxiliary problem can be introduced by considering the homogeneous problem which the solution of temperature distribution satisfies homogeneous government equations (3)-(6) with $p(x, y, t) = 0$. As stated in [7], the auxiliary problem can be the following Sturm-Liouville problem with specific boundary conditions.

$$\nabla^2 \phi_{ilq}(x, y, z) + \lambda_{ilq}^2 \phi_{ilq}(x, y, z) = 0; \; (x, y, z) \in D \tag{8}$$

$$\left.\frac{\partial \phi_{ilq}(x, y, z)}{\partial x}\right|_{x=0, L_x} = \left.\frac{\partial \phi_{ilq}(x, y, z)}{\partial y}\right|_{y=0, L_y} = 0 \tag{9}$$

$$\kappa \left.\frac{\partial \phi_{ilq}(x, y, z)}{\partial z}\right|_{z=-L_z} = h_p \phi_{ilq}(x, y, -L_z) \tag{10}$$

$$\kappa \left.\frac{\partial \phi_{ilq}(x, y, z)}{\partial z}\right|_{z=0} = h_s \phi_{ilq}(x, y, 0) \tag{11}$$

The solutions of Sturm-Liouville problem form a set of completely ortho-normal bases in the spatial domain of die, and their general form can be found in [7]. By applying the general form into our problem, setting h_s to be zero (This assumption is only for comparing our method with [6] under the same experimental setting. Our solver can easily take into account the effect of second heat flow path.), and with several manipulations, $\phi_{ilq}(x,y,z)$ can be got as

$$\phi_{ilq}(x,y,z) = \frac{\cos(\frac{i\pi x}{L_x})\cos(\frac{l\pi y}{L_y})\cos(\lambda_{z_q} z)}{\sqrt{N_{ilq}}}, \tag{12}$$

where N_{ilq}=$\theta_{il}L_xL_yN_{z_q}$, θ_{00}=1/2, θ_{i0}=θ_{0l}=1/4, θ_{il}=1/8 with i≠0 and l≠0, N_{z_q}=L_z+κ/h_p×$\sin^2(\lambda_{z_q}L_z)$, and κ/h_p×λ_{z_q}=cot $(\lambda_{z_q}L_z)$.

Those solutions, $\phi_{ilq}(x,y,z)$'s, are called eigenfunctions. Each of them has a corresponding non-zero eigenvalue, λ^2_{ilq}, and N_{ilq} is the normalized value. Each eigenvalue is equal to

$$\lambda^2_{ilq} = \lambda^2_{x_i} + \lambda^2_{y_l} + \lambda^2_{z_q}, \tag{13}$$

where $\lambda^2_{x_i}$=$(i\pi/L_x)^2$ and $\lambda^2_{y_l}$=$(l\pi/L_y)^2$. The $\lambda^2_{x_i}$, $\lambda^2_{y_l}$, $\lambda^2_{z_q}$ are eigenvalues in x-, y-, and z-directions, and λ_{z_q} can be solved by applying the Newton-Raphson method [9].

Essentially, the equivalent thermal conductivity of second heat flow path should not be zero. Setting h_s to be zero which is the same as [6] is only for comparing our method with [6] under the same experimental setting. The effect of second heat flow path can be easily taken into account in our solver because the general solution of Sturm-Liouville problem already takes h_s into account. Thus, only the computational formulas of N_{z_q} and λ_{z_q} are needed to be modified, and this will not influence the derivation of computational formula and evaluating algorithm of the temperature in the remaining portion of this work.

B. System Transformation for Time-Varying Coefficients

Since the generated bases are completely ortho-normal in the spatial domain of die, $T(x,y,z,t)$ can be approximated by $\widehat{T}(x,y,z,t)$ as

$$\widehat{T}(x,y,z,t) = \sum_{q=0}^{N_z-1}\sum_{l=0}^{N_y-1}\sum_{i=0}^{N_x-1}\psi_{ilq}(t)\phi_{ilq}(x,y,z), \tag{14}$$

where $\psi_{ilq}(t)$ is the time-varying coefficient, N_x, N_y, and N_z are truncation points in x-, y-, and z-directions, respectively.

Here, our major goal is to find an analytical expression of $\psi_{ilq}(t)$ for achieving an accurate temperature approximation. Substituting equation (14) into (3), the residual function $r(x,y,z,t)$ is equal to

$$r(x,y,z,t) = \kappa\nabla^2\epsilon(x,y,z,t) - \sigma\frac{\partial\epsilon(x,y,z,t)}{\partial t}, \tag{15}$$

where $\epsilon(x,y,z,t) = T(x,y,z,t) - \widehat{T}(x,y,z,t)$.

To accurately approximate $T(x,y,z,t)$ by equation (14), the norm of $r(x,y,z,t)$ should be as small as possible. In order to achieve this goal, the following steps are performed to find the desired expression of $\psi_{ilq}(t)$. Due to the limitation of space, we only list the derived steps of $\psi_{ilq}(t)$ in brief and the detail derivation is ignored.

- Expand $r(x,y,z,t)$ by using $\phi_{ilq}(x,y,z)$ as

$$r(x,y,z,t) = \sum_{q=0}^{\infty}\sum_{l=0}^{\infty}\sum_{i=0}^{\infty} r_{ilq}(t)\phi_{ilq}(x,y,z), \tag{16}$$

  where $r_{ilq}(t)$=$\int_{-L_z}^{0}\int_0^{L_y}\int_0^{L_x} r(x,y,z,t)\phi_{ilq}(x,y,z)dxdydz$.
- Perform the Galerkin's scheme [8] which sets the $r_{ilq}(t)$'s to be zeros up to truncation points N_x, N_y, and N_z.
- Transfer the resulted equation to the form which preserves the law of conservation by using Divergence Theorem [7].
- Apply equation (8) and the ortho-normality of $\phi_{ilq}(x,y,z)$'s to the resulted equation for getting the following un-coupled system.

$$\sigma\psi'_{ilq}(t) = -\kappa\lambda^2_{ilq}\psi_{ilq}(t) + \widehat{p}_{ilq}(t), \text{ and } \psi_{ilq}(0) = 0; \tag{17}$$

  where $\widehat{p}_{ilq}(t) = \int_0^{L_y}\int_0^{L_x} p(x,y,t)\phi_{ilq}(x,y,0)dxdy$, $0 \le i \le N_x$, $0 \le l \le N_y$, and $0 \le q \le N_z$,
- Obtain the general solution of each $\psi_{ilq}(t)$'s as following.

$$\psi_{ilq}(t) = \frac{1}{\sigma}\int_0^t \widehat{p}_{ilq}(\tau)e^{-\frac{k}{\sigma}\lambda^2_{ilq}(t-\tau)}d\tau. \tag{18}$$

Equation (18) can be applied to obtain the steady-state temperature without any time step evaluation, and equation (17) is applied to obtain the transient temperature distribution.

C. Average Temperature Evaluation of Grid Cells

Generally, the hot spots occur in the regions closing to power sources. Hence, we focus on evaluating the average temperature of each grid cell on the top surface (z=0) of die. First, we present the formulation for calculating the average steady-state temperature and analyze its convergent property. Then, the fast evaluating algorithms are developed for realizing the formulation. Finally, the transient analysis is given.

C.1. Steady State Formulation and its Convergence Rate Analysis

When calculating the steady-state temperature distribution, each turning on/off function of grid cell is a unit step function. Hence, the close-form of each $\psi_{ilq}(\infty) = \widehat{p}_{ilq}(\infty)/(k\lambda^2_{ilq})$ can be analytically obtained from equation (18) without any time step evaluation. After that, plugging $\phi_{ilq}(x,y,z)$'s and $\psi_{ilq}(\infty)$'s into equation (14), the average steady state rising temperature, $\overline{T}_{mn}$, of grid cell (m,n) on the top surface is

$$\begin{aligned}\overline{T}_{mn} &= \frac{1}{\Delta x\Delta y}\int_{n\Delta y}^{(n+1)\Delta y}\int_{m\Delta x}^{(m+1)\Delta x}\widehat{T}(x,y,0,\infty)dxdy \\ &= \sum_{l=0}^{N_y-1}\sum_{i=0}^{N_x-1}K_{il}\cos\left(\frac{i\pi(2m+1)}{2M}\right)\cos\left(\frac{l\pi(2n+1)}{2N}\right)\end{aligned} \tag{19}$$

where

$$K_{il} = \frac{\widehat{P}_{il}}{\kappa}\sum_{q=0}^{N_z-1}\frac{C_{ilq}}{N_{ilq}}, \tag{20}$$

$$C_{ilq} = \begin{cases} \frac{\Delta x\Delta y}{\lambda^2_{ilq}}; & i=0, l=0 \\ \frac{4NL_y\Delta x\sin^2\left(\frac{l\pi}{2N}\right)}{l^2\pi^2\lambda^2_{ilq}}; & i=0, l\neq 0 \\ \frac{4ML_x\Delta y\sin^2\left(\frac{i\pi}{2M}\right)}{i^2\pi^2\lambda^2_{ilq}}; & i\neq 0, l=0 \\ \frac{16MNL_xL_y\sin^2\left(\frac{i\pi}{2M}\right)\sin^2\left(\frac{l\pi}{2N}\right)}{i^2l^2\pi^4\lambda^2_{ilq}}; & i\neq 0, l\neq 0 \end{cases} \tag{21}$$

and

$$\widehat{P}_{il} = \sum_{m=0}^{M-1}\sum_{n=0}^{N-1}p_{mn}\cos\left(\frac{i\pi(2m+1)}{2M}\right)\cos\left(\frac{l\pi(2n+1)}{2N}\right). \tag{22}$$

Before introducing our evaluating algorithms for calculating the average steady state rising temperature, we first proceed the convergent analysis to show the benefit of our temperature calculating formula. The error of our temperature calculating formula can be bounded by the following theorem.

Theorem 1: The absolute truncation error of average steady state temperature for each grid cell (m,n) by using the GIT based formulation with truncation points N_x, N_y, and N_z in x-, y-, and z-directions is bounded by

$$\sum_{(i,l,q)\in S_1}\frac{\alpha_1}{i^2l^2\lambda^2_{ilq}} + \sum_{(i,q)\in S_2}\frac{\alpha_2}{i^2\lambda^2_{i0q}} + \sum_{(l,q)\in S_3}\frac{\alpha_3}{l^2\lambda^2_{0lq}} + \sum_{q\in S_4}\frac{\alpha_4}{\lambda^2_{00q}}, \tag{23}$$

where S_1=$[1, N_x] \times (N_y, \infty) \times [0, \infty) \cup (N_x, \infty) \times [1, N_y] \times [0, \infty) \cup [1, N_x] \times [1, N_y] \times (N_z, \infty)$, S_2=$[1, N_y] \times (N_z, \infty) \cup (N_x, \infty) \times [0, \infty)$, S_3=$[1, N_x] \times (N_z, \infty) \cup (N_y, \infty) \times [0, \infty)$, S_4=(N_z, ∞), α_1=$128M^2N^2P_T/(L_xL_yL_z\kappa\pi^4)$, α_2=$16M^2P_T/(L_xL_yL_z\kappa\pi^2)$, α_3=$16N^2P_T/(L_xL_yL_z\kappa\pi^2)$, *and* α_4=$2P_T/(L_xL_yL_z\kappa)$.

Due to the limitation of space, the proof is ignored. The above result shows that the decaying rate of the truncation error of our GIT based method can be in the order of $i^2l^2((i\pi/L_x)^2 + (l\pi/L_y)^2 + \lambda_{z_q}^2)$.

To compare the convergent rate, we also obtain the truncation error bound of formulation in [6] as following.

$$\sum_{(i,l)\in B_1} \frac{\beta_1}{i^2l^2\gamma_{il}} + \sum_{i\in B_2, l=0} \frac{\beta_2}{i^2\gamma_{il}} + \sum_{i=0, l\in B_3} \frac{\beta_3}{l^2\gamma_{il}}, \tag{24}$$

where γ_{il}=$\sqrt{(i\pi/L_x)^2 + (l\pi/L_y)^2}$, B_1=$(N_x, \infty) \times (N_y, \infty)$, $B_2 = (N_x, \infty)$, B_3=(N_y, ∞), β_1=$64M^2N^2P_T/(L_xL_y\kappa\pi^4)$, $\beta_2 = 8M^2P_T/(L_xL_y\kappa\pi^2)$, and $\beta_3 = 8N^2P_T/(L_xL_y\kappa\pi^2)$.

This bound shows that the decaying rate of the truncation error of Green's function based formulation is in the order of $i^2l^2\sqrt{(i\pi/L_x)^2 + (l\pi/L_y)^2}$.

Therefore, the convergence rate of GIT based method is much faster than Green's function based method [6]. The reason is that the GIT based method generates the ortho-normal spatial bases for the *transient heat diffusion equation*, and obtains the close-form of steady state solution by using these spatial bases which can fully fill the eigen-space of heat diffusion equation. On the other hand, [6] constructs the spatial approximated function by applying Green's function to Possion's equation which does not contain the temporal information. As a result, the generated Green's function could not fully fill the eigen-space of transient heat diffusion equation for approximating the temperature.

The convergence rate of our GIT based formulation is not only faster than [6], the experimental results also demonstrate that it can maintain the same accuracy as [6] even if the truncation point, N_x or N_y, is far less than the number of divisions, M or N.

Although the truncation points N_xN_y can be far less than the number of grid cells MN, there is no actual efficiency improvement over [6] if we directly apply the standard FFT to evaluate each T_{mn} because the standard FFT need pad zeros to the input data when the dimensions of input and output data are not equal. To overcome this limitation, we provide fast evaluating algorithms for our GIT formulation without the zero padding.

C.2. Fast Evaluating Algorithms for GIT Formulation

To efficiently realize our formulation of steady state temperature distribution, we first derive a one-dimensional radix-two based FFT like evaluating algorithm for the length of output data being larger than the length of input data, *1D-STL-FFT*. Then, based on *1D-STL-FFT*, we develop another one-dimensional FFT like evaluating algorithm for the length of output data being smaller than the length of input data, *1D-LTS-FFT*. Finally, these two algorithms are integrated to calculate equations (19) and (22) by the row-column procedure, and the computational complexity of our GIT based thermal simulator can be analyzed to be only $\underline{O(MN\log_2 N_xN_y)}$.

a) 1D-STL-FFT: The prototype of *1D-STL-FFT* is

$$\overline{F}_k = \sum_{i=0}^{N_x-1} f_i e^{j2\pi ik/2M}; \quad k = 0, \cdots, 2M-1, \tag{25}$$

where N_x<M and each is power of 2, j=$\sqrt{-1}$, and f_i's and $\overline{F}_k$'s are complex input and output data, respectively. Since the length of $\overline{F}_k$'s is larger than the length of f_i's, the zeros-padding step of f_i's used in standard FFT algorithm for evaluating $\overline{F}_k$'s should be avoided to save runtime. Therefore, our *1D-STL-FFT* algorithm, a modified FFT algorithm, is developed as stated in Fig 2 to calculate Equation (25) without zeros-padding. Firstly, the "**Reverse-bit**(f)" performs $\log_2 N_x$ times of the Danielson-Lanczos lemma [9] to Equation (25) and reorders the input data. Then, the *1D-STL-FFT* algorithm evaluates the output of those L sub Discrete Fourier Transforms (DFTs) in the bottom level by using *Line* 3∼16, and performs *Line* 17 to get the output of rest levels.

```
Algorithm Radix-two 1D-STL-FFT
Input: Complex vector $f$ with length $N_x$
Output: Complex vector $\overline{F}$ with length $2M$
1    Begin
2      $f_{\mathbf{R}}$ = Reverse-bit($f$) ;
3      $L = 4M/N_x$ ;
4      $N_{SubDFTs} = N_x/2$ ;
5      For $SubIndex = 0$ to $N_{SubDFTs} - 1$
6        $k = L \times SubIndex$ ;
7        $i = 2 \times SubIndex$ ;
8        For $SubK = 0$ to $L - 1$
9          $\overline{F}[k] = \overline{F}[k] + f_{\mathbf{R}}[i] + f_{\mathbf{R}}[i+1] \times e^{j2\pi \times SubK/L}$ ;
10         $k = k + 1$ ;
11       EndFor
12     EndFor
13     Apply the bottom up procedure of standard FFT to
       execute $\log_2 N_x - 1$ times Danielson-Lanczos lemma
       of (25) for evaluating the final $\overline{F}$
14   End
```

Fig. 2. Procedure of *1D-STL-FFT*. The "**Reverse-bit**" means the reverse-bit algorithm [9]

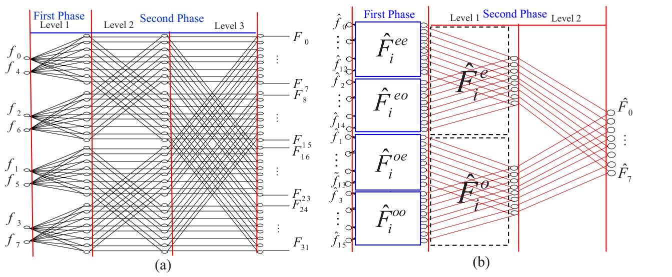

Fig. 3. Computational flow graphs of *1D-STL-FFT* and *1D-LTS-FFT* with $N_x = 8$ and $M = 16$. (a) The *1D-STL-FFT*. (b) The *1D-LTS-FFT*.

An example with $M = 16$ and $N_x = 8$ is given in Fig. 3.(a), there are 3 bisecting levels, and 4 sub DFTs in the bottom level. After performing the reverse-bit algorithm to input data, two phases are executed. The first phase is done by using *Line* 3∼16 of Fig. 2. Then, the second phase is to get output of rest levels by executing the bottom up procedure of standard FFT as stated in *Line* 17 of Fig. 2.

The complexity of *1D-STL-FFT* is $O(M\log_2 N_x)$ because there are $\log_2 N_x$ bisecting levels and the complexity of each level is $O(M)$.

b) 1D-LTS-FFT: The prototype of *1D-LTS-FFT* is

$$\widehat{F}_i = \sum_{m=0}^{M-1} \widehat{f}_m e^{j2\pi im/2M}; \quad i = 0, \cdots, N_x - 1, \tag{26}$$

where $M > N_x$, and $\widehat{F}_i$ and $\widehat{f}_m$ are complex output and real input data, respectively. Repeating the Danielson-Lanczos lemma with $\log_2(M/N_x)+1$ times, $\widehat{F}_i$ can be written as the sum of $2M/N_x$ sub DFTs, and each sub DFT has the same form as the *1D-STL-FFT* with input and output length are $N_x/2$ and N_x, respectively.

Two phases are utilized to evaluate $\widehat{F}_i$ and the *1D-LTS-FFT* algorithm is summarized in Fig. 4. First, *Line* 2 performs the reverse-bit algorithm to the input data, and *Line* 4∼8 use the *1D-STL-FFT* algorithm to obtain each bisected sub DFT. After each sub DFT being done, a bottom up procedure is applied to the rest $\log_2(M/N_x)+1$ bisecting levels for finding $\widehat{F}_i$ and the executing steps are from *Line* 9∼26.

```
Algorithm Radix-two 1D-LTS-FFT
Input: Real vector $\widehat{f}$ with length $M$
Output: Complex vector $\widehat{F}$ with length $N_x$
1  Begin
2      $\widehat{f}_{\mathbf{R}}$ = Reverse-bit($\widehat{f}$) ;
3      $N_{SubDFTs} = 2M/N_x$ ;
4      For $Sub_i$ = 0 to $N_{SubDFTs} - 1$
5          $Start = Sub_i \times N_x$ ;
6          $End = Start + N_x$;
7          $F_t(Start : End - 1)$ = 1D-LTS-FFT$\Big(\widehat{f}_{\mathbf{R}}(\frac{Start}{2} : \frac{End}{2} - 1)\Big)$ ;
8      EndFor
9      $L = N_x$ ;
10     For $level$ = 0 to $log_2(M/N_x)$
11         $Next^*$ = 0 ;
12         $Sub_i$ = 0 ;
13         $N_{SubDFTs}$ = $N_{SubDFTs}$ / 2 ;
14         While $Sub_i < N_{SubDFTs}$
15             For $i$ = 0 to $N_x - 1$ ;
16                 $b1 = i + Sub_i \times N_x$ ;
17                 $b2 = b2 + N_x$ ;
18                 $n = i + Next^*$ ;
19                 $F_t[n] = F_t[b1] + F_t[b2] \times e^{j2\pi i/L}$ ;
20             EndFor
21             $Sub_i = Sub_i + 2$ ;
22             $Next^* = Next^* + N_x$ ;
23         EndWhile
24         $L = 2 \times L$ ;
25      EndFor
26      $\widehat{F} = F_t(0 : N_x - 1)$;
27 End
```

Fig. 4. Procedure of *1D-LTS-FFT*.

An example with $M = 16$ and $N_x = 8$ is shown in Fig. 3.(b). In the first phase, the input data are reordered by using the reverse-bit algorithm, and these reordered data are fed into the corresponding *1D-STL-FFT* blocks. This can be done by using *Line* 3∼8 in Fig. 4. Then, the output of top block in the level 1 of second phase is calculated by

$$\widehat{F}_i^e = \widehat{F}_i^{ee} + e^{j2\pi i/16}\widehat{F}_i^{eo}, \tag{27}$$

and $\widehat{F}_i^o$ can be calculated by using a similar way. Finally, $\widehat{F}_i$ is equal to

$$\widehat{F}_i = \widehat{F}_i^e + e^{j2\pi i/32}\widehat{F}_i^o. \tag{28}$$

The above computational flow of the second phase is summarized from *Line* 9∼26 in Fig. 4.

For the general case, the sub DFTs in each level of the second phase can be obtained by combining those sub DFTs of their previous level with the similar formula of equation (27) by replacing 16 to be $2N_x$, $4N_x$, $\cdots$, $2M$ in each level. The computational complexity of first phase is $O(M\log_2 N_x)$ because the *1D-STL-FFT* need to be executed $2M/N_x$ times, and each complexity is $O(N_x \log_2 N_x)$. The complexity is $O(M)$ for the second phase. Hence, the computational complexity of *1D-LTS-FFT* is $O(M\log_2 N_x)$.

c) Temperature Evaluation: The average steady state rising temperature, $\overline{T}_{mn}$, shown in equation (19), can be evaluated as

$$\overline{T}_{mn} = \frac{1}{2}R_e\left\{\overline{F}_{m,n} + \overline{F}_{2M-(m+1),n}\right\}, \tag{29}$$

where $R_e\{\cdot\}$ is the real part operator, and

$$\overline{F}_{k_1,k_2} = \sum_{i=0}^{N_x-1}\sum_{l=0}^{N_y-1}\overline{K}_{il}e^{\frac{j2\pi ik_1}{2M}}e^{\frac{j2\pi lk_2}{2N}}. \tag{30}$$

Here, $0 \leq k_1 \leq 2M-1$, $0 \leq k_2 \leq 2N-1$, $\overline{K}_{il} = K_{il}e^{j2\pi i/4M}e^{j2\pi l/4N}$, and each K_{il} is equal to equation (20).

In the following, we are going to utilize the *1D-STL-FFT* algorithm to develop a row-column procedure to calculate $\overline{F}_{k_1,k_2}$'s. The K_{il}'s can also be obtained by using a similar procedure with the *1D-LTS-FFT* algorithm. The row-column based *2D-STL-FFT* method for calculating $\overline{F}_{k_1,k_2}$'s is summarized in Fig. 5. *Line* 2∼4 performs *1D-STL-FFT* to each row of the input matrix $\overline{K}$ which each (i,l) entry is equal to $\overline{K}_{il}$, and then *Line* 5∼7 applies

```
Algorithm Radix-two 2D-STL-FFT
Input: Complex matrix $\overline{K}$ with length $N_x \times N_y$
Output: Complex matrix $\overline{F}$ with length $2M \times 2N$
1      Begin
2          For i = 0 to $N_x - 1$
3              $T_{Row}(i, 0 : 2N - 1)$ = 1D-STL-FFT$\Big(\overline{K}(i, 0 : N_y - 1)\Big)$ ;
4          EndFor
5          For j = 0 to $2N - 1$
6              $\overline{F}(0 : 2M - 1, j)$ = 1D-STL-FFT$(T_{Row}(0 : N_x - 1, j))$ ;
7          EndFor
8      End
```

Fig. 5. Procedure of *2D-STL-FFT*.

1D-STL-FFT to each column of output matrix got from the row procedure to obtain the desire matrix $\overline{F}$. Since the complexity of *1D-STL-FFT* is $O(M\log_2 N_x)$, the total complexity of evaluating $\overline{F}_{k_1,k_2}$'s by this row-column procedure is $\underline{O(MN\log_2 N_x)}$.

To obtain each K_{il} from equation (20), $\widehat{P}_{il}$'s need to be known from equation (22). Therefore, the two dimensional type of equation (26) is needed to get related $\widehat{F}_{i,l}$'s for input data being p_{mn}'s. Similarly, a *1D-STL-FFT* based row-column procedure can be used to get those related $\widehat{F}_{i,l}$'s. However, the form of equation (29) can not be utilized to calculate $\widehat{P}_{il}$'s because the lengths of those related $\widehat{F}_{i,l}$'s in row and column directions are less than $2M$ and $2N$, respectively. Therefore, the complex conjugates of $\widehat{F}_{i,l}$'s are required to complete the calculation of $\widehat{P}_{il}$'s.

Fortunately, the complex conjugate of the output from each sub *1D-STL-FFT* in calculating $\widehat{F}_{i,l}$'s can be directly obtained by reversing these sub DFTs , for example, $(\widehat{F}_i^{ee})^* = \widehat{F}_{N_x-i}^{ee}$ in Fig. 4. Therefore, the complex conjugate of $\widehat{F}_{i,l}$'s can be got by firstly reversing the data of F_t from *Line* 7 in Fig. 4, and performing *Line* 9∼26 in Fig. 4 during the row-column procedure of $\widehat{F}_{i,l}$'s.

Similar to the analysis of $\overline{F}_{k_1,k_2}$'s, the complexity for evaluating $\widehat{F}_{i,l}$'s is $O(MN\log_2 N_y)$. The complexity of calculating the negative frequency components of $\widehat{F}_{i,l}$'s is $O(MN) + O(N_yN)$ since only the second phase need to be recomputed. Therefore, the complexity for computing equation (22) is $\underline{O(MN\log_2 N_y)}$.

¿From the above discussion, we conclude that the complexity of our GIT based thermal simulator is $\underline{O(MN\log_2 N_xN_y)}$.

C.3. Transient Simulation

To perform the transient simulation, the turning on/off function of each grid, $H_{mn}(t)$, is a time interval function specified by instruction. After applying finite difference schemes (For simplicity, we use the backward-Euler method.) to equation (17), each time-varying coefficient, ψ_{ilq}^t, at the sampling time t is

$$\psi_{ilq}^t = \frac{\sigma}{R_{ilq}}\psi_{ilq}^{t-\Delta t} + \frac{\Delta t}{R_{ilq}\sqrt{N_{ilq}}}\widehat{P}_{il}^t, \tag{31}$$

where Δt is the time step, $R_{ilq} = \sigma + \kappa\lambda_{ilq}^2\Delta t$, $\widehat{P}_{il}^t$ is equal to equation (22) with p_{mn} replaced by $p_{mn}H_{mn}^t$, and H_{mn}^t is the value of turning on/off function at time step t. After time-varying coefficients at time t being calculated, the average temperature in each grid cell at time step t can be obtained by equation (14) with the same evaluating method presented in previous subsection.

IV. Experimental Results

We implement our GIT based thermal simulator and the Algorithm II of a highly efficient Green's function based method [6] in C++ language. The state-of-the-art FFT package, FFTW3 [10], is used to realize the DCT and IDCT for [6]. All methods are tested on a HP xw9300 workstation with 16 GB memory. The results are compared with a commercial computational fluid dynamic software, ANSYS®.

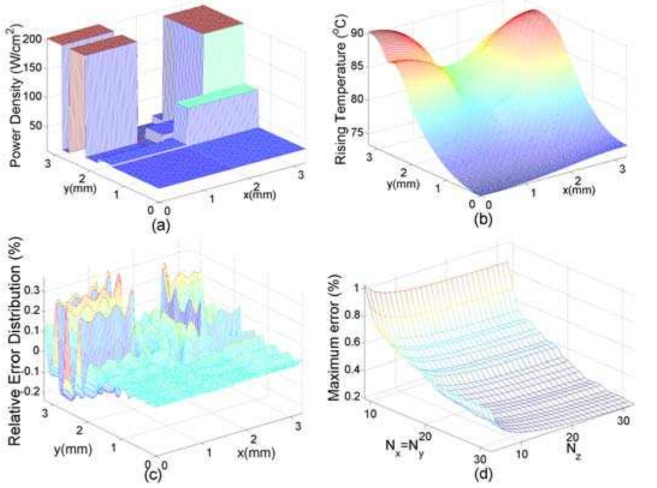

Fig. 6. Accuracy and the maximum error trend of proposed GIT based method. (a) The power density map of test chip. (b) The estimated temperature distribution of test chip. (c) The relative error distribution of GIT based method compared with the result of ANSYS®. Here, the numbers of truncation points are 16×16×70. (d) The maximum relative error versus the numbers of truncation points.

A. Accuracy and Fast Convergence of the GIT Based Thermal Simulator

The chip, DEC Alpha 21264, is employed to demonstrate the accuracy of our method, and its size is scaled down to 3.3mm×3.3mm×0.5mm. Its power density map is shown in Fig. 6.(a), and the power sources are on the top surface of functional blocks. The equivalent heat transfer coefficient of primary heat transfer path is 8700 W/($\mathrm{m}^2\cdot{}^\circ\mathrm{C}$), and the thermal conductivity of silicon is 148 W/($\mathrm{m}\cdot{}^\circ\mathrm{C}$). The above settings are the same as [6]. The power density map is divided into 128×128 grid cells. The average steady state temperature distribution on the top surface of die computed by our GIT based method with the truncation point being 16 in x-, y-direction, and 70 in z-direction is shown in Fig. 6.(b). The maximum relative error compared with the result of ANSYS® is 0.3732%, and its relative error distribution is shown in Fig. 6.(c). On the other hand, the truncation point of Green's function based method [6] need to be 2048 in both x- and y-directions that its number of bases is 234 times larger than our method to achieve the same accuracy level, and its maximum relative error is 0.3735%. This reveals the fast convergence advantage of our proposed GIT based method. To further demonstrate our fast convergence rate, we plot the maximum relative errors with different truncation points in Fig 6.(d). As you can see that our GIT based analyzer can achieve an extremely accurate solution even when the truncation points are very small.

B. Thermal Simulation for Full-Chip Containing Lots of Functional Blocks

To demonstrate the capability of our GIT based method for the thermal simulation of full-chip with containing lots of functional blocks, and the efficiency improvement of our GIT based method over the Algorithm II of Green's function based method [6], we consider a test chip with dimension 1cm×1cm×0.5mm. It consists of one million functional blocks, the power density of each block is between 3.0e4 W/m^2 and 1.5e6 W/m^2, and its power density map is illustrated in Fig. 7.(a). The top surface of this chip is discretized into 1024×1024 square grid cells. The truncation points of proposed method are set to be 16×16×8 and the truncation points of [6] are set to be 2048×2048 to achieve the similar maximum error. The temperature distribution of top surface calculated by the proposed method is shown in Fig. 7.(b), and its maximum error is only 0.3576% presented in TABLE I.

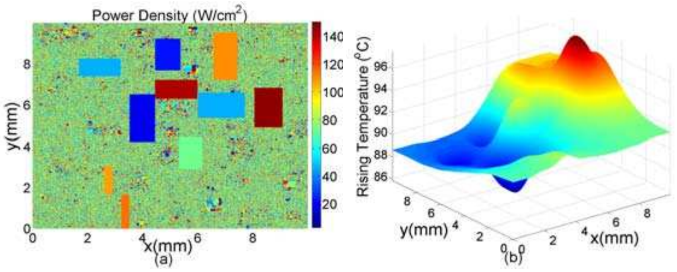

Fig. 7. The power density and temperature distribution of a 1cm×1cm chip with one million functional blocks. (a) The power density map. (b) The estimated temperature distribution.

The runtime comparison is shown in TABLE I. The run-time of post-calculating stage in our method is only 0.1312 seconds while the run-time of post-calculating stage in [6] is 2.7642 seconds. The speedup of our method over the Algorithm II of [6] is 21.07 at the post-calculating stage. This result demonstrates the dramatic efficiency improvement of our thermal analyzer over [6].

		Green's function based [6]	GIT based
number of functional blocks		1 million	
number of grid cells		2^{20}	
number of bases		2^{22}	2^{11}
max error (%)		0.4143	0.3576
runtime (s)	pre-calculating	2.4785	0.00005
	post-calculating	2.7642	0.1312
speedup (post-calculating)		21.0686	

TABLE I
COMPARISON OF THE PROPOSED GIT BASED METHOD AND ALGORITHM II OF [6] FOR A CIRCUIT WITH ONE MILLION FUNCTIONAL BLOCKS.

V. CONCLUSIONS

An accurate and efficient GIT based thermal simulator has been presented. Experimental results confirm its theoretical property which can achieve extremely accurate results with sufficiently small truncation points. The proposed algorithm only takes 0.13 seconds for the thermal analysis of full chip with one million functional blocks and over one million grid cells in the post-calculating stage to achieve accurate steady state temperature distribution. Therefore, the proposed GIT based thermal simulator is very suitable for the thermal-aware design flow. Finally, the early-stage 3-D chip thermal analysis can be achieved by numerical schemes and combining our proposed analytical technique in this paper, and this will be our future work.

REFERENCES

[1] T. -Y. Wang and C. C. -P. Chen, "Thermal-ADI: A Linear-Time Chip-Level Thermal Simulation Algorithm Based on Alternating-Direction Implicit (ADI) Method," in *TVLSI*, vol. 11, no. 4, pp. 691-700, Aug. 2003.

[2] T. -Y. Wang and C. C. -P. Chen, "SPICE-Compatible Thermal Simulation with Lumped Circuit Modeling for Thermal Reliability Analysis Based on Model Reduction," in *ISQED*, pp. 357-62, Mar. 2004.

[3] P. Li, L. T. Pileggi, M. Asheghi, and R. Chandra, "IC Thermal Simulation and Modeling via Efficient Multigrid-Based Approaches," in *TCAD*, vol. 25, no. 9, pp. 319-26, Sep. 2006.

[4] W. Huang, S. Ghosh, S. Velusamy, K. Sankaranarayanan, K. Skadron and M. R. Stan "HotSpot:ACompact Thermal Modeling Methodology for Early-Stage VLSI Design," in *TVLSI*, vol. 14, no. 5, pp. 501-13, May 2006.

[5] J.-L. Tsai, C. C.-P. Chen, G. Chen, B. Goplen, H. Qian, Y. Zhan, S.-M. Kang, M. D. F. Wong and S. S. Sapatnekar, "Temperature-Aware Placement for SOCs," in *Proceedings of the IEEE*, vol. 94, no. 8, pp. 1502-18, Aug. 2006.

[6] Y. Zhan, and S. S. Sapatnekar, "High Efficiency Green Function-Based Thermal Simulation Algorithms," in *TCAD*, accepted for future publication.

[7] M. D. Mikhailov, and M. N. Ozisik, "Unified Analysis and Solutions of Heat and Mass Diffusion," John Wiley & Sons Inc., NY, 1983.

[8] M. D. Mikhailov, "General Solutions of the Diffusion Equations Coupled at the Boundary Conditions," in *Int. J. Heat Mass Transf.*, vol. 16, pp. 2155-64, 1973.

[9] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, "Numerical Recipes in C++," Cambridge Unvi. Press, 2002.

[10] M. Frigo and S. G. Johnson, "FFTW version 3.1 package," in http://www.fftw.org.