A Fast Leakage Aware Green's Function Based Thermal Simulator for 3D Chips

Hameedah Sultan and Smruti R Sarangi

***Abstract*—In this paper we propose a fast thermal modeling tool, *3DSim*, using a Green's function based approach. Green's function based approaches have been shown to be faster than the traditional finite difference based techniques. Our proposed tool can model steady state and transient thermal profiles for both 2D and 3D chips, which may contain multiple active layers and fluid-carrying microchannels for heat removal. The unique advantage of our tool is that it models leakage power analytically using a piecewise linear leakage model, thereby eliminating the need to iterate multiple times through the leakage-temperature feedback loop. We use several algebraic techniques and transforms to compute the thermal profile analytically and thereby speed up the process of temperature calculation. To the best of our knowledge transform based approaches have not been used before to model the temperature in 3D chips with microchannels. Our approach provides a 150X speedup over state of the art thermal simulators, with an error limited to 5%.**

***Index Terms*—Thermal simulation, Hankel transform, 3D chips, microchannels, leakage, power, chip level, Green's function.**

I. Introduction

In modern day chips, temperature is a first order design constraint. In order to mitigate the effects of high temperature and to make the design thermally aware, a fast method to estimate the chip temperature is necessary at various stages of the design process. For example, at the level of architectural exploration it is necessary to perform extensive thermal simulations to figure out an optimal layout of cores and cache banks with the limited information that is available. In the placement and floorplanning stages, temperature simulations are often carried out to ensure that hotspots do not form. Finally, after the entire synthesis process is over simulations are carried out to get an accurate estimate of the on-chip temperature for various workloads. For all of these use cases it is necessary to conduct hundreds of temperature simulations, and thus the speed of simulation without a concomitant loss in accuracy is *essential*. To the best of our knowledge ultra-fast approaches based on Green's functions for estimating temperature in 3D chips with microchannels has been an open problem till now.

3D chips have been proposed to reduce the latency between processor and memory using embedded DRAM technologies, and to increase the number of logic layers in a package (increased computational density). In 2D chips, the heat spreader and the heat sink serve to limit the temperature rise. Sadly, in 3D chips, the dissipation of heat through the spreader and heat sink is not very effective because of the presence of multiple layers. As the number of layers increases, there exist layers farther from the heat sink whose heat cannot be effectively removed by traditional air cooling. Also the power density is higher in 3D chips. Consequently, temperature becomes a limiting factor. Several approaches have been suggested to remove the heat generated in 3D chips. These include etching fluid carrying microchannels under each power dissipating layer, thermo-electric cooling, and intelligent via placement. The most promising solution out of these is to dissipate the excess heat in 3D chips by incorporating fluid-carrying microchannels [1]. Sadly to model the temperature profile of such complicated scenarios using the traditional finite difference based approach is difficult. We need to take into account the physics of heat transfer, fluid mechanics, and the effect of the leakage-temperature feedback loop. Hence, in this paper our aim is to use the ultra-fast Green's function based approaches that have hitherto been shown to work in the case of 2D chips.

The Green's function is defined as the impulse response (thermal profile) of a unit power source. Once the impulse response has been obtained, the temperature profile for any given power profile can be obtained by simply convolving the Green's function with the power profile. Green's function based methods are significantly faster than traditional finite difference or finite element based methods. This is because unlike the finite difference method, where the solution has to be obtained for the entire heat transfer path, in these approaches it is just sufficient to take the power dissipating layers in to account and furthermore convolution is a faster operation than matrix multiplication. Additionally, the grid size in the finite difference approach has to be small enough, such that the derivative can be replaced by a difference operation across adjacent grids, whereas the grid size in the Green's function approaches is the resolution at which the output thermal profile is needed, usually the size of functional units. This reduces the complexity of the problem enormously, resulting in a significant speedup.

Along with computing the temperature of a 3D-chip we take the leakage-temperature feedback loop into account as well. Computing leakage is far more difficult in 3D chips because different layers influence each other. Most thermal simulators [2], [3], [4] follow an iterative approach (compute temperature $\rightarrow$ compute leakage $\rightarrow$ recompute temperature $\rightarrow$ ...), which is time consuming. Moreover, this process has prohibitive overheads wile computing transient thermal profiles. Sadly, it is not possible to ignore this step because according to Huang et al. [5], ignoring leakage power can

Hameedah Sultan is with the School of Information Technology, Indian Institute of Technology, Delhi, India, 110016.
E-mail: anz158222@cse.iitd.ac.in
Smruti R Sarangi is with the Department of Computer Science and Engineering, Joint Professor in Electrical Engineering, Indian Institute of Technology, Delhi, 110016.
E-mail: srsarangi@cse.iitd.ac.in
Manuscript received Month dd, 2020; revised Month dd, 2020.

lead to a temperature estimation error of at least 4-7°C for a processor at the 130 nm technology node.

Finally, we simulate the effect of microchannels as well. Other than the work by Yan et al. [6], none of the other 3D thermal simulators model leakage power and microchannels. Yan et al. [6] use an iterative solver based on the multigrid method, which is far slower than our approach. The reason our approach is fast is because *we propose closed-form analytical solutions that incorporate the effects of both leakage and microchanels*. We do not need multiple iterations.

Our main contributions are briefly summarized as follows:

1) We propose an analytical framework for computing the leakage aware thermal profile for both steady state and transient temperature estimation in 3D chips.
2) Our approach is based on Green's functions, and hence it is at least 150X faster than state of the art numerical approaches.
3) Our simulator analytically accounts for leakage power using several approximations and complex transforms. For improved modeling accuracy, it assumes a piece-wise linear model of leakage.
4) We introduce a new way of adapting the proposed method for anisotropic systems, such as for a chip with microchannels. We also propose a set of corrections to obtain an accurate estimate of the edge and corner effects that arise in 3D chips with microchannels.

We compare our results with a commercial CFD software, Ansys Icepak, as well as state of the art open-source tools (HotSpot [7] and 3D-ICE [3]).

The rest of the paper is organized as follows. In Section II we provide the relevant background. In Section III we describe our basic thermal modeling framework. In Section IV we adapt our method to account for the anisotropicity in microchannels, and complete this by describing our piecewise linear leakage modeling approach in Section V. In Section VI we evaluate our algorithm and compare it with other state of the art methods. We describe the related work in Section VII and then conclude in Section VIII.

II. Background

A. Power Dissipation in a Chip

1) Dynamic Power: Dynamic power is dissipated when switching activity happens on a chip. In older technologies, dynamic power used to be the dominant cause of power dissipation, but in newer technologies, leakage power is becoming increasingly significant.

2) Leakage Power: The subthreshold leakage power, P_{leak}, is exponentially dependent on temperature and is given by the simplified BSIM 4 [8] model:

$$P_{leak} \propto v_T^2 * e^{\frac{V_{GS}-V_{th}-V_{off}}{\eta * v_T}}(1 - e^{\frac{-V_{DS}}{v_T}}) \quad (1)$$

where, v_T is the thermal voltage (kT/q), V_{th} is the threshold voltage, V_{off} is the offset voltage in the sub-threshold region and η is a constant.

However, using an exponential model in thermal modeling is very expensive. Moreover, the operating range of real ICs usually stays limited to 40 - 80°C . In [9], it was shown that the error in leakage power computed using a linear model stays limited to 10%, which translates to a 1.33% error in temperature. Using a piecewise linear leakage model further reduces the error to 0.33%. Similar results were obtained experimentally in [10], [11], [12]. Hence leakage power is always modeled by a linear or a piecewise linear approximation in the thermal modeling domain [6], [10], [11], [13]. We too use a linear leakage power model to begin with, given by Equation 2.

$$P_{leak} = P_{leak0} + \beta(T - T_{amb}), \quad (2)$$

where, T_{amb} is the ambient temperature, P_{leak0} is the leakage power at ambient temperature, and $\beta = \frac{dP_{leak}}{dT}$ is a function of the electrical characteristics of the chip such as threshold voltage, and supply voltage.

To further improve upon the accuracy, we then modify our proposed method for a **piecewise linear leakage** model.

Let us now understand the equations governing the heat transfer in chips.

B. Thermal Modeling

1) Fourier Heat Equation: Heat transfer in a chip is governed by the Fourier's law of heat conduction:

$$\nabla.(\kappa \nabla T) + \dot{q} = C_v \frac{\partial T}{\partial t}, \quad (3)$$

where T is the temperature, C_v is the volumetric specific heat, and t represents time.

In the case of microchannels, the heat equation for the fluid layer is given by:

$$\nabla \cdot (k \nabla T) + \dot{q} = C_v \frac{\partial T}{\partial t} + C_v \vec{u}.\nabla T, \quad (4)$$

where $\vec{u}$ is the fluid velocity along the microchannel.

Traditional thermal simulators solve this differential equation using the finite difference method, and invoke the analogy between thermal and electrical circuits to convert this into an RC circuit. The second term on the right hand side arises because of the fluid flow; it is modeled as a voltage controlled current source in finite difference based approaches [3]. In that case, the heat transfer between the solid and fluid layers is modeled by a conductance, g, whose value is given by:

$$g = h_f \times l \times w, \quad (5)$$

where h_f is the vertical heat transfer coefficient, and l and w are the length and width of the microchannel layer respectively. The vertical heat transfer coefficient, h_f, is given by:

$$h_f = \frac{k_{fluid} \times Nu}{d_h}, \quad (6)$$

where, k_{fluid} is the conductivity of the fluid, Nu is the Nusselt number, which is equal to the ratio of the convective and conductive heat transfer across the solid-fluid boundary, and d_h is the hydraulic diameter = $\frac{2hl}{h+l}$. (h is the height of the microchannel layer.) A higher Nusselt number indicates more active convection. The Nusselt number has known empirical forms for a variety of geometries. For forced convection, it is a function of the Reynolds number (a dimensionless number governing the transition between laminar and turbulent flow) and the Prandtl number (another dimensionless number which is constant for a fluid and a fluid state).

In this paper, instead of relying on the slower finite difference or finite element methods, we shall use the much faster Green's function based approach.

2) Green's Function: Instead of discretizing the heat equation, it is also possible to analytically solve it for a set of boundary conditions when an impulse power source is applied as the input. The solution thus obtained is the *Green's function*. The temperature profile corresponding to any arbitrary input is then given by the convolution of the Green's function with the power profile (distribution of power within a chip) as follows:

$$\mathcal{T} = G \star P \tag{7}$$

Here, $\mathcal{T}$ is the temperature profile (change in temperature), $\star$ is the 2-D convolution operator, G is the Green's function, and P is the power map.

The **main focus of our technique is the speed of computation while considering leakage power**. The method of re-iterating till convergence consumes a lot of time, especially for transient analysis. Zhou et al. [14] show that ignoring temperature dependent leakage in modern-day chips can result in a 32% error. Furthermore, they also show that **a 20% reduction in peak temperature can be obtained by simply modeling temperature dependent leakage during the floorplanning phase without any overheads. However, it increases the floorplanning run time by 56%.** Hence, often where speed is of utmost importance, researchers either ignore leakage, or make very crude approximations, compromising heavily on accuracy. Thus there is a strong need for a fast algorithm that can provide the steady state as well as the transient temperature profile while incorporating the effects of leakage in a modern 3D chip with microchannels.

We consequently devised a novel set of algebraic techniques to compute the leakage aware Green's function in a 3-D chip. We also propose novel approximations for sides and corners. We then use a combination of several mathematical techniques to provide an analytical solution to a complex set of equations. Finally, we compare our results against those obtained using commercial CFD software, and competing approaches.

III. Methodology

TABLE I: Glossary

Symbol	Meaning
l	Active silicon layers in the chip
n	Number of grids along any direction
fsp_{il}	Green's function for layer i when the source is in layer l
fsp_{il}^{L}	Leakage aware Green's function for layer i when the source is in layer l
fsp_{il}^{res}	Residual Green's function for layer i when the source is in layer l
$\mathcal{T}$	Temperature rise above the ambient temperature
β	Temperature dependence of leakage power
t	Time
h	Hankel domain variable
s	Laplace domain variable

A. Model of the Chip

We assume that the 3D chip has l active layers. Beneath each active layer, there is a microchannel and a thermal interface material (TIM) layer. We discretize each layer into an $n \times n$ grid. The complete model of the chip is shown in Figure 1a for a chip with a heat sink. The chip with microchannels is shown in Figure 1b, and a glossary of the terms used is presented in Table I.

B. Overview of our Thermal Modeling Approach

Green's function based thermal modeling involves three main steps [2]:

1) Obtain the Green's functions.
2) Convolve the Green's functions with the power profile.
3) Apply corrections to the edges and corners. The side walls of a chip are adiabatic in nature, which prevents heat from flowing across these boundaries. Ziabari et al. [2] use the method of images from electromagnetics to account for the adiabatic boundaries. They extend the power matrix by padding it with zeros making it three times its original size, and then place image sources (a source on the other side of the boundary at the same distance) across the boundaries.

We propose to augment this process by introducing three new steps, and modifying the procedure to account for the edge and corner effects:

1) After obtaining the Green's functions, we then modify them by solving a set of equations to obtain the leakage aware Green's functions for both the steady state and the transient case.
2) In the case of anisotropic systems such as chips with microchannels, directly using the Green's functions obtained in Step (1) leads to large errors in the downstream regions[1] of the chip. We propose the use of a *residual Green's function*, which removes these errors.
3) We also propose a modified approach to account for edges and corners in chips with microchannels. Here, the upstream and downstream boundaries along the flow of the fluid are no longer adiabatic in nature, since the fluid carries heat along with it. As a resulted, the complete heat is not reflected back at the boundaries, rather it gets damped exponentially. Hence, we place image sources across the boundaries and convolve it with the modified Green's functions that are damped by an exponential function.
4) Finally, we propose a framework for leakage modeling using piecewise-linear models.

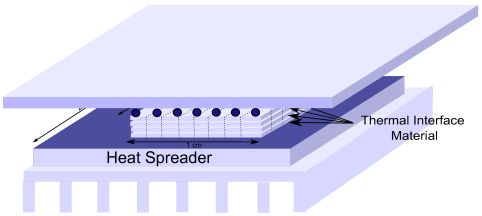
An overview of our approach is shown in Figure 1c.

In this section, we describe our thermal modeling approach for a homogeneous chip-heatsink structure without microchannels. In Section IV, we shall adapt this approach to accommodate for the anisotropicity introduced by microchannels. We shall then move on to describe our piecewise linear thermal modeling approach in Section V.

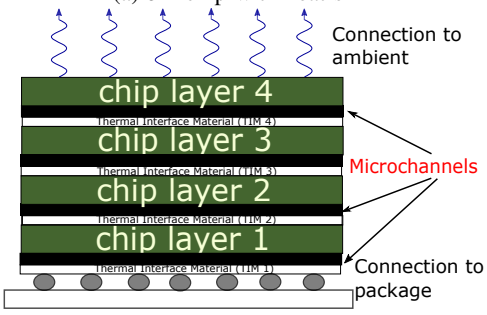
C. Obtaining the Green's Functions without Leakage

The Green's functions can either be theoretically calculated or obtained from thermal sensors in the chip, or estimated using a thermal simulator. Theoretical computation of the Green's functions is very complex, because of the presence of multiple layers, and the interaction between conductive and convective heat transfer. A lot of assumptions have to be made to make a theoretical computation possible, which leads to large errors in temperature estimation [2]. As a result, most modern Green's function based thermal simulators obtain the Green's functions empirically [2], [11], [12], since this needs

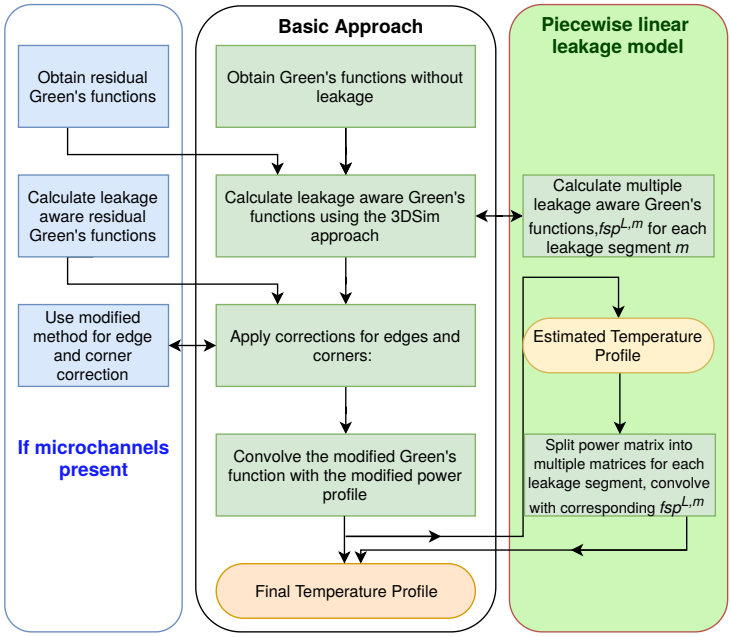
[1] A microchannel flows from upstream regions to downstream regions



(a) 3D chip with heat sink



(b) 3D chip with microchannels



(c) Overview of our approach

Fig. 1: Model of the chip and our modeling approach

to be done only once for a given geometry. We too adopt the same approach. The two empirical approaches are either direct measurement using thermal sensors, or using slow FEM based thermal simulators.

One might argue that the leakage aware Green's functions can directly be obtained and stored. However, obtaining the leakage aware functions takes a long time (of the order of several hours) using FEM (or FDM) simulators, and the process is extremely expensive for transient temperature computation. Even if we slightly vary the parameters such as the supply voltage, operating frequency, or the threshold voltage, we will have to recompute these functions. Studying the effects of varying these parameters on power and performance is a very popular approach in computer architecture and EDA research. Hence, computing leakage aware Green's functions using conventional techniques is *not practical*. An additional benefit of a fast analytical method for calculating these functions is that we can simulate on-chip process variation and resultant leakage power variation. The computation time for these tasks using conventional software is prohibitive.

D. 3D Leakage Aware Green's Function

1) Steady State Solution: When a dynamic power source, P_{dyn} is applied to a 2D chip, a corresponding leakage power P_{leak} is dissipated. The final temperature rise $\mathcal{T}$ is given by:

$$\mathcal{T} = f_{sp} \star (P_{dyn} + \Delta P_{leak}), \tag{8}$$

where f_{sp} is the heat spreading function or the Green's function.

Now, let us assume we have an l-layer chip. We apply a unit impulse source at the center of layer k. We will have a total of l^2 Green's functions, corresponding to impulse sources applied to layers 1 through l. The Green's functions or the heat spreading function in layers 1 to l are denoted by fsp_{1k}, fsp_{2k}, ... fsp_{lk}. fsp_{ik} denotes the effect in layer i, when a point source is applied in layer k. We denote the corresponding temperature rise in layer i by $\mathcal{T}_{ik}$. $\mathcal{T}_{ik}$ is affected by the dynamic power dissipated by layer k, as well as the leakage sources created in all the layers. We assume a linear leakage model for the time being, $\Delta Pleak_{ik} = \beta \mathcal{T}_{ik}$. This corresponds to the leakage sources created in layer i due to the dynamic source in layer k. We extend Equation 8 to an l-layer chip:

$$\begin{aligned}\mathcal{T}_{1k} =& fsp_{11} \star \beta\mathcal{T}_{1k} + fsp_{12} \star \beta\mathcal{T}_{2k} + ... \\ &+ fsp_{1k} \star (P_{dyn} + \beta\mathcal{T}_{kk}) + ... + fsp_{1l} \star \beta\mathcal{T}_{lk} \\ &...\end{aligned}$$

Since P_{dyn} is a delta function and the convolution of any function with a delta function is the function itself, we arrive at the following set of equations:

$$\begin{aligned}\mathcal{T}_{1k} =& fsp_{1k} + \beta(fsp_{11} \star \mathcal{T}_{1k} + fsp_{12} \star \mathcal{T}_{2k} + fsp_{13} \star \mathcal{T}_{3k} \\ &+ ... + fsp_{1l} \star \mathcal{T}_{lk}) \\ &...\end{aligned}$$

Now, to simplify the convolution operations and replace them with multiplications, we compute the 2-D Fourier transform on both sides. After simplifying, we obtain a set of simultaneous linear equations in $\mathcal{F}(\mathcal{T}_{1k})$, $\mathcal{F}(\mathcal{T}_{2k})$, ... $\mathcal{F}(\mathcal{T}_{lk})$, where $\mathcal{F}$ denotes the Fourier transform operator. To solve this system of equations, we used Mathematica [15]. The number of terms in the resulting equation is very large. To reduce the number of terms, we ignore all second and higher order leakage terms, i.e. terms where the exponent of β is greater than one.

We rearrange the terms to arrive at an analytical solution, which can be used directly to obtain the final temperature profile 9. We call this method *3DSimF*.

$$\mathcal{F}(\mathcal{T}_{12}) = \frac{\begin{array}{r}\mathcal{F}(fsp_{12}) + \beta(\mathcal{F}(fsp_{13})\mathcal{F}(fsp_{32}) - \mathcal{F}(fsp_{33})\mathcal{F}(fsp_{12}) \\ + ... + \mathcal{F}(fsp_{1l})\mathcal{F}(fsp_{l2}) - \mathcal{F}(fsp_{ll})\mathcal{F}(fsp_{12}))\end{array}}{1 - \beta\left(\mathcal{F}(fsp_{11}) + \mathcal{F}(fsp_{22}) + ... + \mathcal{F}(fsp_{ll})\right)} \tag{9}$$

After computing the inverse Fourier transform of Equation 9, we can obtain the leakage aware Green's function. However, to calculate the final temperature profile, we would again need to compute its inverse transform. Hence we store our result in the transform domain itself.

Thus we divide the calculation of leakage aware Green's functions into two sub-stages. In the first sub-stage we compute the transform of the heat spreading functions. In the second sub-stage we use Equation 9 to calculate the leakage aware Green's functions in the transform domain. The advantage of this approach is that whenever β changes (as a result of varying the supply voltage or the threshold voltage, or by performing voltage-frequency scaling), we will have to re-run only the second sub-stage, that is, the calculation of the final Green's functions.

Utilizing symmetry to reduce an $O(n^2)$ problem to an $O(n)$ problem: The method discussed above ($3DSimF$) requires computing a 2-D transform, which is computationally expensive. To further speed-up the computation we describe another variant of our approach, $3DSimH$, that uses the Hankel transform ($\mathcal{H}$).

A 1-D zero order Hankel transform is equivalent to a 2-D Fourier transform for a radially symmetric function. The Hankel transform uses Bessel functions as its basis functions. Thus the 2-D problem ($O(n^2)$ points) is reduced to a single dimension ($O(n)$ points). The Hankel transform is defined as:

$$\mathcal{H}(f(r)) = H(h) = \int_0^\infty f(r)\mathcal{J}_0(hr)rdr \tag{10}$$

Here $\mathcal{J}_0$ is a Bessel function of the first kind of order 0, h is the Hankel transform variable and $\mathcal{H}$ denotes the Hankel transform operator.

Using the Hankel transform in place of the Fourier transform in Equation 9 we arrive at:

$$\begin{array}{l}\mathcal{H}(\mathcal{T}_{12}) = \\ \frac{\begin{array}{r}\mathcal{H}(fsp_{12}) + 2\pi\beta(\mathcal{H}(fsp_{13})\mathcal{H}(fsp_{32}) - \mathcal{H}(fsp_{33})\mathcal{H}(fsp_{12}) \\ + ... + \mathcal{H}(fsp_{1l})\mathcal{H}(fsp_{l2}) - \mathcal{H}(fsp_{ll})\mathcal{H}(fsp_{12}))\end{array}}{1 - 2\pi\beta\left(\mathcal{H}(fsp_{11}) + \mathcal{H}(fsp_{22}) + ... + \mathcal{H}(fsp_{ll})\right)} \\ ...\end{array} \tag{11}$$

However, this method does not apply to chips with microchannels since the thermal profile is no longer radially symmetric. Thus for chips with microchannels, $3DSimF$ will have to be used. The rest of the approach remains the same as in $3DSimF$. We collectively refer to both of these variants as *3DSim*, and differentiate only when necessary.

2) Transient Solution: To calculate the transient profile, we add a capacitive term, which captures the temperature rise with time (as done in [11]). Without any loss of generality, let us assume that the source is present in layer 2, and the chip has four active layers.

$$\begin{array}{l}\mathcal{T}_{12} = fsp_{12} + \beta\left(fsp_{11}\star\mathcal{T}_{12} + fsp_{12}\star\mathcal{T}_{22} + fsp_{13}\star\mathcal{T}_{32} + fsp_{14}\star\mathcal{T}_{42}\right) - \\ C_1\left(fsp_{11}\star\frac{\partial T_{12}}{\partial t} + fsp_{12}\star\frac{\partial T_{22}}{\partial t} + fsp_{13}\star\frac{\partial T_{32}}{\partial t} + fsp_{14}\star\frac{\partial T_{42}}{\partial t}\right) \\ ...\end{array} \tag{12}$$

We note that the non-capacitive terms on the right hand side of Equation 12 comprise the steady state response. Let us call this $\mathcal{T}_{12ss}$ ($ss \rightarrow$ steady state). We first compute the Hankel transform to convert the convolution operations to multiplication. This results in a set of differential equations. We then compute the Laplace transforms in the time domain. This enables us to convert the differential equations to algebraic equations and we get a set of four linear equations. We finally arrive at:

$$\begin{array}{l}\mathcal{L}(\mathcal{H}(\mathcal{T}_{12})) = \mathcal{L}\left(\mathcal{T}_{12ss}\right) - \frac{2\pi C_1}{1 - 2\pi\beta fsp_{11}} \times \\ \left(\mathcal{H}(fsp_{11})\mathcal{L}\left(\mathcal{H}\left(\frac{\partial T_{12}}{\partial t}\right)\right) + \mathcal{H}(fsp_{12})\mathcal{L}\left(\mathcal{H}\left(\frac{\partial T_{22}}{\partial t}\right)\right) + \right. \\ \left. \mathcal{H}(fsp_{13})\mathcal{L}\left(\mathcal{H}\left(\frac{\partial T_{32}}{\partial t}\right)\right) + \mathcal{H}(fsp_{14})\mathcal{L}\left(\mathcal{H}\left(\frac{\partial T_{42}}{\partial t}\right)\right)\right) \\ ...\end{array} \tag{13}$$

where $\mathcal{L}$ is the Laplace transform operator.

Using properties of the Laplace transform, setting the temperature rise at $t = 0$ to be zero, and solving further, we arrive at Equation 14.

$$\begin{array}{l}s\mathcal{L}(\mathcal{H}(\mathcal{T}_{12})) = \mathcal{T}_{12ss} - s^2\frac{2\pi C_1}{1 - 2\pi\beta fsp_{11}}\left(\mathcal{H}(fsp_{11})\mathcal{L}(\mathcal{H}(T_{12})) + \right. \\ \left. \mathcal{H}(fsp_{12})\mathcal{L}(\mathcal{H}(T_{22})) + \mathcal{H}(fsp_{13})\mathcal{L}(\mathcal{H}(T_{32})) + \mathcal{H}(fsp_{14})\mathcal{L}(\mathcal{H}(T_{42}))\right) \\ ...\end{array} \tag{14}$$

where s is the Laplace transform variable.

Let $f_{11} = \frac{2\pi C_1}{1-2\pi\beta fsp_{11}}$, which is constant across time.

The ideal approach would be to solve the set of equations in Equation 14 for $\mathcal{L}(\mathcal{H}(\mathcal{T}_{12}))$, calculate the inverse Laplace transform, followed by the inverse Hankel transform. However, the complexity involved quickly makes the problem intractable, even using numerical techniques.

Hence, we ignore the terms that have a minimal contribution to the computed temperature field. We only incorporate the first and second order effects produced by the current and adjoining layers (ignore tertiary effects). This gives us two capacitive terms for each layer:

$$\begin{array}{l}s\mathcal{L}(\mathcal{H}(\mathcal{T}_{12})) = \mathcal{T}_{12ss} - s^2 f_{11}\left(\mathcal{H}(fsp_{11})\mathcal{L}(\mathcal{H}(T_{12})) + \mathcal{H}(fsp_{12})\mathcal{L}(\mathcal{H}(T_{22}))\right) \\ ...\end{array}$$

Solving these, we get,

$$\mathcal{L}(\mathcal{H}(\mathcal{T}_{12})) = \frac{\mathcal{T}_{12ss} + sf_{22}\mathcal{H}(fsp_{22})\mathcal{T}_{12ss} - sf_{11}\mathcal{H}(fsp_{12})\mathcal{T}_{22ss}}{\begin{array}{l}s(-1 - f_{11}\mathcal{H}(fsp_{11})s - f_{22}\mathcal{H}(fsp_{22})s + \\ f_{11}f_{22}\mathcal{H}(fsp_{12})\mathcal{H}(fsp_{21})s^2 - f_{11}f_{22}\mathcal{H}(fsp_{11})\mathcal{H}(fsp_{22})s^2)\end{array}}$$

Next we ignore the secondary feedback effects, (all terms except the first and third terms in the denominator, since the source is in layer 2), and calculate the inverse Laplace transform:

$$\mathcal{H}(\mathcal{T}_{12}) = \mathcal{H}(\mathcal{T}_{12ss}) - \frac{\mathcal{H}(\mathcal{T}_{22ss})f_{11}\mathcal{H}(fsp_{12})e^{\frac{-t}{f_{22}\mathcal{H}(fsp_{22})}}}{f_{22}\mathcal{H}(fsp_{22})} \tag{15}$$

Now, to calculate the final leakage aware transient Green's function, we compute the inverse Hankel transform of Equa-

tion 15:

$$\mathcal{T}_{12} = \mathcal{T}_{12ss} - finv, \text{ where}$$
$$finv = \mathcal{H}^{-1}\left(\frac{\mathcal{H}(\mathcal{T}_{22ss})f_{11}\mathcal{H}(fsp_{12})e^{-t/(f_{22}\mathcal{H}(fsp_{22}))}}{f_{22}\mathcal{H}(fsp_{22})}\right)$$
$$...$$
$$finv = \mathcal{H}^{-1}\left(\frac{\mathcal{H}(\mathcal{T}_{2ss})C_1(1-2\pi\beta\mathcal{H}(fsp_{22}))\mathcal{H}(fsp_{12})e^{\frac{-t}{f_{22}\mathcal{H}(fsp_{22})}}}{C_2(1-2\pi\beta\mathcal{H}(fsp_{11}))\mathcal{H}(fsp_{22})}\right) \tag{16}$$

This equation is still too complex to be calculated analytically. To compute $finv$, we divide the range $(0,\infty)$ into two parts - $(0,\epsilon)$ and (ϵ,∞). We further note that at lower frequencies ($h < \epsilon$),

$$finv_0^{\epsilon} = \int_0^{\epsilon} \frac{C_1}{C_2}\mathcal{H}(\mathcal{T}_{22ss})e^{-\frac{t}{f_{22}\mathcal{H}(fsp_{22})}}\mathcal{J}_0(hr)hdh \tag{17}$$

where $finv_0^{\epsilon}$ is the value of $finv$ between 0 and ϵ.

Since $\epsilon \to 0$, the product $hr \to 0$, and thus $\mathcal{J}_0(hr) \to 1$. At $h \to \epsilon$, $f_{22} = f_{220} = \frac{2\pi C_2}{1-2\pi\beta\mathcal{H}(fsp_{22})|_{h=0}}$,

Also, since $\delta(h)/h \gg \beta\mathcal{T}$ when $h < \epsilon$, we can ignore all the leakage terms. We can thus approximate $\mathcal{H}(\mathcal{T}_{22ss}) = \mathcal{H}(f_{sp22}) = \mathcal{H}(fsp_{22})|_{h=0}\delta(h)/h$. Thus we have,

$$\begin{aligned} finv_0^{\epsilon} &= \int_0^{\epsilon} \mathcal{H}(fsp_{22})|_{h=0}\frac{\delta(h)}{h}e^{-\frac{t}{f_{220}\mathcal{H}(fsp_{22})|_{h=0}\frac{\delta(h)}{h}}}hdh \\ &= \frac{(\mathcal{H}(fsp_{22})|_{h=0})^2 f_{220}}{\epsilon^2 t}\left(1 - e^{\frac{-t\epsilon^2}{f_{220}\mathcal{H}(fsp_{22})|_{h=0}}}\right) \end{aligned} \tag{18}$$

Next we compute $finv$ between (ϵ,∞). We note that the ratio of the terms $\frac{(1-2\pi\beta\mathcal{H}(fsp_{22})\mathcal{H}(fsp_{12})}{(1-2\pi\beta\mathcal{H}(fsp_{11})\mathcal{H}(fsp_{22})}$ is close to 1, when $h > \epsilon$, (where h is the Hankel transform variable, and $\epsilon \to 0$). So we replace this by a correction factor equal to 1.1 (based on empirical results).

$$finv_{\epsilon}^{\infty} = \mathcal{H}^{-1}\left(1.1 \times \frac{\mathcal{H}(\mathcal{T}_{22ss})C_1 e^{-t/(f_{22}\mathcal{H}(fsp_{22}))}}{C_2}\right) \tag{19}$$

where $finv_{\epsilon}^{\infty}$ is the value of $finv$ between ϵ and ∞. As h increases, $finv_0^{\epsilon} \to 0$, and $finv_{\epsilon}^{\infty}$ will dominate. The final transient Green's function can be found using Equation 20.

$$\begin{aligned} \mathcal{T}_{12} &= \mathcal{T}_{12ss} - finv_0^{\epsilon} - finv_{\epsilon}^{\infty} \\ &... \end{aligned} \tag{20}$$

E. Convolution with the Power Profile

In the online stage of our algorithm, these leakage aware Green's functions are convolved with the power map to obtain the leakage aware full chip thermal profile. Since we have saved the leakage aware Green's functions in the transform domain itself, we compute the Fourier transform of the power source, multiply them and take the inverse Fourier transform. For a 3D chip, the effects of sources in all layers have to be accumulated. Thus we use Equation 21.

$$\begin{aligned} T_i = \mathcal{F}^{-1}\Big(&\mathcal{F}(fsp_{i1}^L)\mathcal{F}(\mathcal{P}_1) + \mathcal{F}(fsp_{i2}^L)\mathcal{F}(\mathcal{P}_2) + ...+ \\ &\mathcal{F}(fsp_{il}^L)\mathcal{F}(\mathcal{P}_l)\Big) \end{aligned} \tag{21}$$

where, $\mathcal{P}_i$ represents the dynamic sources in layer i, fsp_{i1}^L represents the leakage aware Green's function for layer i because of a source in layer l and T_i represents the complete temperature rise in layer i.

Full Chip Transient Thermal Profile: To compute the full chip transient thermal profile, we can use an approach similar to Ziabari et al. [2], after substituting the leakage aware Green's function in place of the basic Green's function. Each convolution operation is also replaced by multiple convolution terms as per Equation 21. The basic idea here is that at any time instant t, we need to convolve the power profile of the last k time instants with the corresponding Green's functions and add them up. Here k is the number of time intervals beyond which the impulse response falls below 5% of its peak value.

Alternatively, we propose another approach that relies on the step response rather than the impulse response.

Instead of modeling the power profile as a series of delta functions, we model them using *step functions*. At any time instant, we subtract the power value in the previous time instant from the power for the current time instant. This difference maybe negative as well, signalling a decay in temperature. The difference is then convolved with a slice of the leakage aware step response corresponding to that time instant. At any time instant, we do this for the last k time steps and add them to get the temperature profile for that time instant. After k time steps, the thermal contribution of the power value at time $t-k$ (=$P(t-k) \star fsp(t-k)$) would saturate to its steady state value. Thus if we replace $fsp(t-k)$ with the steady state value, we do not need to add the effects of sources beyond k time steps. In practice, we slightly overestimate the temperature beyond k ms until the corresponding power sources actually achieve steady state. However this error is limited to 5%, and vanishes very quickly.

F. Corrections for Edges and Corners

An important aspect of Green's function based approaches is the modeling of adiabatic boundaries accurately. For this reason, we use the method of images (similar to Ziabari et al. [2]). We extend the power matrix to twice its original size by padding it with zeros. Next we replace the adiabatic boundaries with image sources on the edges – for every source x units away from the boundary, we place a source $-x$ units away from the boundary (on the other side) . The extended power matrix is then convolved with the leakage aware Green's functions.

G. Techniques used for Further Speed-up

A continuous Hankel transform requires the calculation of the Bessel function for each value of r. To further speed-up the simulation, we use the technique proposed by Johnson [16] to compute the discrete Hankel transform (Equation 22):

$$\mathcal{H}(j_{0,k}) \approx \frac{1}{{j_{0,N+1}}^2}\sum_{n=1}^{N}\frac{2}{J_1^2(j_{0,n})}f(j_{0,n}/j_{0,N+1})J_0(j_{0,k}j_{0,n}/j_{0,N+1}) \tag{22}$$

where $j_{0,k}$ is the k^{th} root of the Bessel function, and N is the defined range of the function whose Hankel transform is to be computed. To avoid computing the roots of the Bessel function in each iteration, we pre-compute the roots and store them in a look-up table.

IV. Modified Thermal Modeling Approach for Microchannels

In this section, we describe the modifications that we make to our basic approach in order to account for anisotropic systems. Although we describe our method for microchannels, our approach can accommodate the anisotropicity introduced by through-silicon-vias.

The heat spreading function is no longer isotropic (same in all directions) in the presence of microchannels, as can be seen in Figure 2. To account for the anisotropicity, we propose the use of a *residual Green's function*, which we describe next.

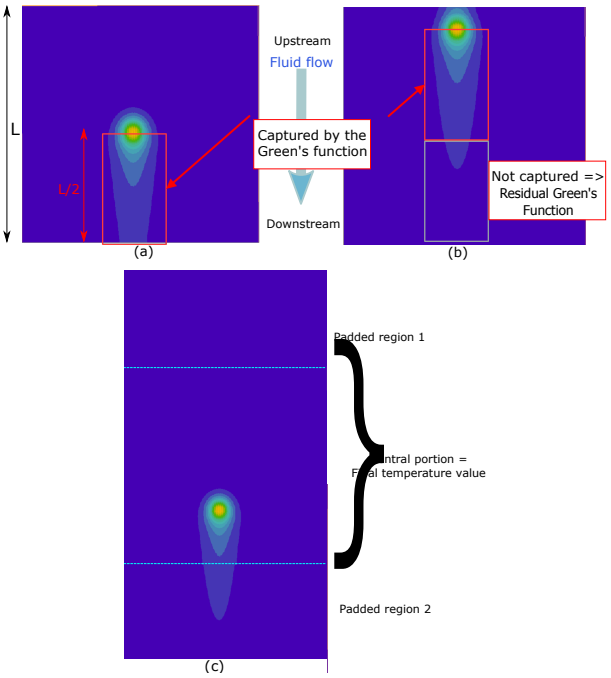


Fig. 2: Handling the anisotropicity in the Green's function

A. Residual Green's Functions for Chips with Microchannels

In the case of anisotropic systems such as chips with microchannels, the Green's function is skewed in one direction because the fluid carries heat along its flow, resulting in a higher temperature rise downstream (Figure 2, where we assume the fluid to flow from the upper (upstream) region to the bottom (downstream region)). Consequently the temperature rise does not drop to zero at the downstream boundary. However, the Green's function (conventionally obtained by applying a power to the center of the chip) captures the thermal information only till the edge of the chip, i.e. till a distance of $L/2$ from the power source. Here L is the length of the chip. Thus the thermal information captured by the Green's function is incomplete. So to capture the complete information, we modify the conventional Green's function approach in the following manner.

We apply a source to the upper (upstream) edge of the chip. The resultant temperature rise is shown in Figure 2(b). To capture the complete thermal information, we consider two functions: the conventional Green's function (source at the center), and an additional *residual Green's function*. The *residual Green's function* is obtained by measuring the temperature rise along the complete microchannel when the source is present at the upstream edge of the chip, and removing the thermal information already present in the conventional Green's function (source at the center). We will now describe this process mathematically. For a unit power source applied at the edge, the temperature rise using the Green's function is given by:

$$\begin{aligned} \mathcal{T}_{12}^{edge} &= fsp_{12} \star \delta(x, y - L/2) \\ &= fsp_{12}(x, y - L/2) \end{aligned} \tag{23}$$

The actual temperature rise obtained is fsp_{12}^{edge}. Thus the *residual Green's function* is given by:

$$fsp_{12}^{res} = fsp_{12}^{edge} - \mathcal{T}_{12}^{edge} \tag{24}$$

When a source is applied in the upstream region (Figure 2(c)), a part of the temperature rise ends up in the zero-padded region (region 2). An additional temperature rise is contributed by the image source (described in more detail in the next section). The central part of the sum of the two temperature profiles is finally extracted, leaving the spurious temperature rise behind in the zero padded region.

To make the *residual Green's function* leakage aware, we use the method described in Section III. While calculating the full chip thermal profile, the *residual Green's function* is also convolved with the power profile and the final thermal profile is obtained by adding it to Equation 21.

B. Corrections for Edges and Corners

We modify the method of images to make it applicable to chips with microchannels (Figure 3). In this method, the adiabatic boundaries are replaced by image sources across the boundary. At the adiabatic chip boundaries, the entire heat spreads back along the silicon layer. However, in chips with microchannels, the boundaries are not completely adiabatic, rather some of the heat is carried away by the microchannels under the chip, and only a part of the heat spreads back along the silicon layer. To account for this effect, we adopt the following approach: We divide the chip into small segments. Some heat is lost to the underlying fluid by each segment lying in the path of the heat spreading back from the boundary. The amount of heat lost by a segment at a distance of r from the edge is given by:

$$\begin{aligned} P_{lost}(r) &\propto \frac{r}{n} \\ &= f\frac{r}{n}, \end{aligned} \tag{25}$$

where f is the constant of proportionality.

Hence the amount of heat that reaches a segment at a distance of r from the edge is given by:

$$P_n(r) = \left(1 - \frac{fr}{n}\right)^n * P_{ref}, \tag{26}$$

where P_{ref} is the amount of heat that would reach this segment if the boundaries were perfectly adiabatic.

As is normally done, let us assume that the length of each segment tends to 0, and thus we have a very large number of such segments. We thus have:

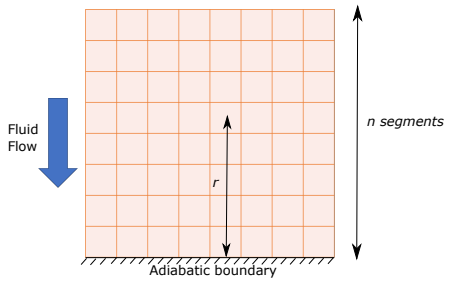


Fig. 3: Edge Effects in Chips with Microchannels

$$P_n(r) = \lim_{n\to\infty}\left(1-\frac{fr}{n}\right)^n * P_{ref} = e^{-fr} * P_{ref} \quad (27)$$

Thus we see that the heat spreading back along the silicon layer is effectively damped by an exponential function.

The image sources on the inlet and outlet edges are convolved with the damped Green's functions, and added to the full chip thermal profile, while the image sources not on the microchannel inlet or outlet edges are convolved with the regular Green's functions.

V. Piecewise Linear Leakage Model

We model the leakage power using an M-segment piecewise model over the entire operating range of the chip (refer to [9]). Let the temperature dependent coefficient of leakage power (β) for each of the M temperature segments be $\beta_0, \beta_1 ..., \beta_{M-1}$. Our piecewise linear leakage temperature computation consists of the following steps:

1) We first compute the leakage aware Green's functions for each of these segments, and store them separately.
2) Next, we compute an approximate thermal profile by convolving the power profile with the Green's functions (assuming a baseline leakage power).
3) Based on the approximate thermal profile obtained, we split the power profile into separate matrices, where each matrix has power numbers corresponding to the locations on the chip where the temperature lies in a particular segment.
4) Finally we convolve each power matrix with the corresponding leakage aware Green's function, and add them to obtain the full chip piecewise leakage aware thermal profile.

An overview of our approach is shown in Figure 4.

We first describe the method by which we obtain the leakage aware Green's functions for each temperature segment. For the sake of better understanding of the proposed solution, we will describe the 2-D piecewise leakage model first, and later extend it to 3D chips.

The temperature rise is given by:

$$\mathcal{T} = fsp \star (P_{dyn} + P_{leak}) \quad (28)$$

Let us assume that the dynamic power consumption is zero initially. The leakage power in the chip is at the baseline level, β_0. Next, we apply a unit power source at the center of the chip. The temperature of the chip rises above the ambient temperature, resulting in a higher value of β at the center.

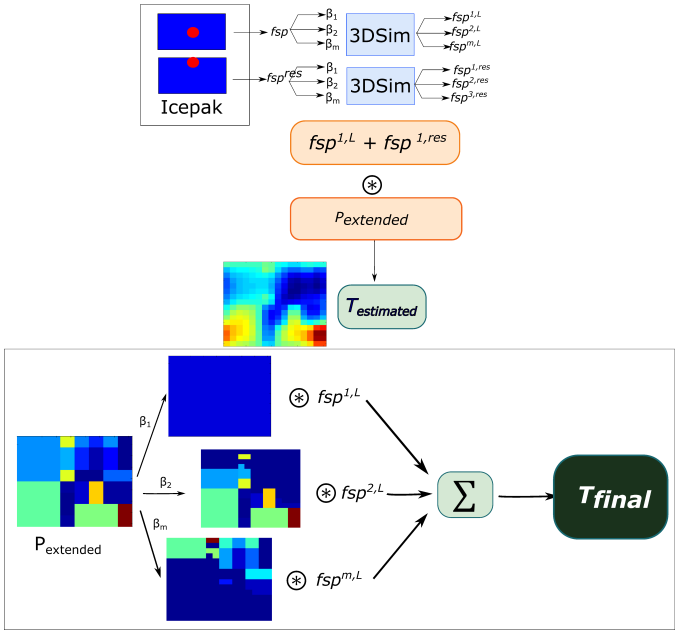


Fig. 4: Overview of our piecewise linear leakage modeling approach

$$\mathcal{T} = fsp + fsp \star (\beta\mathcal{T}) \quad (29)$$
$$= fsp + fsp \star (\beta_0\mathcal{T} + (\beta_1 - \beta_0)\mathcal{T}\delta(x,y)) \quad (30)$$
$$= fsp + fsp \star (\beta_0\mathcal{T} + (\beta_1 - \beta_0)\mathcal{T}|_{x=0,y=0}) \quad (31)$$

Let us denote the maximum temperature rise by $\mathcal{T}_{max}$. Here we make a simplifiying assumption that $fsp|_{x=0,y=0} \approx \mathcal{T}|_{x=0,y=0} = \mathcal{T}_{max}$. Then we have:

$$\mathcal{T} = fsp + fsp \star (\beta_0\mathcal{T} + (\beta_1 - \beta_0)\mathcal{T}_{max}) \quad (32)$$

Taking the Fourier transform on both sides and simplifying, we get:

$$\mathcal{F}(T) = \frac{\mathcal{F}(fsp)(1 + \mathcal{T}_{max}(\beta_1 - \beta_0))}{1 - \beta_0\mathcal{F}(fsp)} \quad (33)$$
$$= \alpha\mathcal{F}(T|_{\beta_0}), \quad (34)$$

where $T|_{\beta_0}$ is the temperature rise when a single segment linear leakage model with coefficient equal to β_0 is used, and $\alpha = (1 + \mathcal{T}_{max}(\beta_1 - \beta_0))$.

Using a similar method, we can derive the Green's functions for a 3D chip:

$$\mathcal{F}(T^m_{i2}) = \alpha^{i,m}\mathcal{F}(T^0_{i2}|_{\beta_0}), \quad (35)$$

where, $\mathcal{F}(T^m_{i2})$ is the piecewise leakage aware Green's function for the m^{th} leakage segment and the i^{th} layer, $\alpha^{i,m} = (1 + \mathcal{T}_{max,i2}(\beta_m - \beta_0))$ and $T^0_{i2}|_{\beta_0}$ is the leakage aware Green's function for the i^{th} layer assuming a baseline leakage power.

Next, we obtain the full chip temperature profile using the method described in Section III-D1. Based on this estimated thermal profile, we divide the power profile into M matrices, where each matrix has power values corresponding to each temperature segment. Finally, we convolve each power matrix with its corresponding leakage aware temperature matrix to obtain the full chip thermal profile.

VI. Evaluation

TABLE II: Parameters of the chip

Parameter	Value
No. of layers, l	4
No. of grid points per layer ,n	16
Die size	100 mm^2
Die thickness	0.15 mm
TIM thickness	0.02 mm
Spreader thickness	3 mm
Microchannel width	50 μm
Microchannel height	100 μm
Prandtl number	0.7085
Fluid velocity	0.75 m/s and 1 m/s

A. Architecture of the 3D Chip

We evaluate our algorithm for two different chips: 3D chip with heat sink and a 3D chip with microchannels (same configuration as [3]).

1) 3D chip with Heat Sink: We consider a 3D chip comprising four active layers of silicon with a layer of thermal interface material in between them. Each layer of silicon has the dimensions: $1cm \times 1cm \times 0.015cm$ and has $16 \times 16 = 256$ grid points (deemed to be enough by [11]). The top layer is attached to a heat spreader which is attached to a heat sink. The top layer of the spreader can be thought of as an isothermal surface (since it is attached to the heat sink), while all other external surfaces of the chip are adiabatic. The parameters of the chip are listed in Table II.

2) 3D chip with Microchannels: The 3D chip with microchannels consists of four active layers of silicon, underneath which there is a microchannel layer and thermal interface layer. The microchannel layer has alternating silicon walls and fluid carrying channels. 100 microchannels of 50 μm each are present that carry water in them. The fluid velocity is constant along the channel. The height of the microchannel layer is 100 μm. We use the same configuration as [3].

B. Setup

We use the commercial CFD simulator, Ansys Icepak, for thermal simulation. It is based on the Fluent CFD package. Multilevel meshing has been used to separately mesh the chip and the spreader.

Our routines are written in R and Matlab for computing and manipulating the Green's functions. We begin by applying a 1 W point power source at the center of the chip at grid point (9, 8) (9^{th} grid point in the x-direction, and the 8^{th} grid point in the y-direction) to obtain the Green's function without leakage from Ansys Icepak. We do this for each layer. Next we employ our proposed methodology to compute the leakage aware Green's functions. To validate our results, we need to calculate the corresponding Green's functions with leakage from Icepak. We iteratively calculate the leakage of each core, apply an equivalent amount of additional power to emulate leakage, and re-run the simulation. All our results have been obtained on an Intel i7 (2.8 GHz) desktop PC with 8 GB RAM running 64-bit Windows 8.

For the transient version, it is extremely difficult to calculate the leakage aware Green's functions using a thermal simulator like Icepak. This is because in order to obtain the leakage converged temperature profile, we need to calculate the additional leakage power based on the temperature at a given time instant, and re-iterate till the power-temperature values converge. In a transient setting, where the temperature changes, we pretty much need steady state (leakage-converged) solutions at each point in time. This is prohibitively expensive. Hence, for validating our results, we do this activity for a fixed number of chosen points in time, and interpolate the results in between.

C. Error Metric

We report the mean absolute error across all layers of a chip, along with the maximum temperature rise. In most cases, we report the percent error relative to the maximum temperature rise of a layer. Other works in temperature estimation often report the error relative to the mean temperature in °C [17], which under-represents the error.

D. Results for 3D Chip with Heat Sink

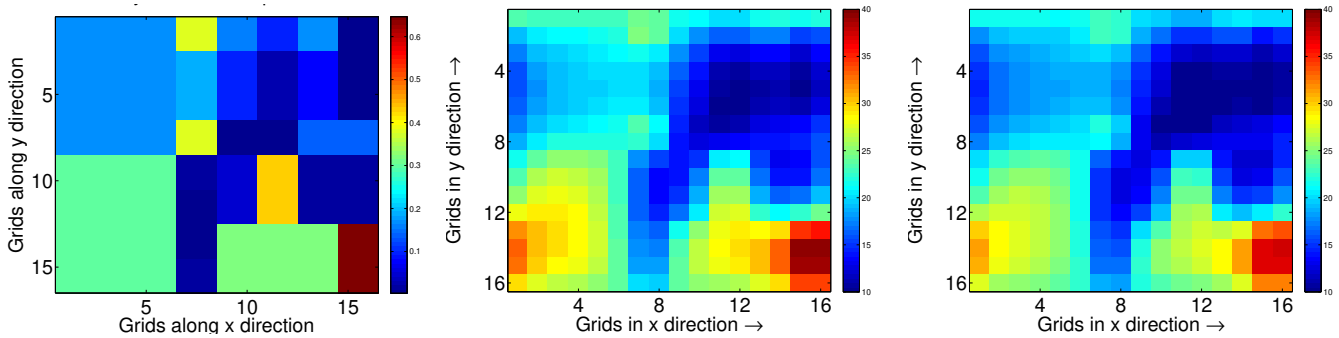
1) Steady State Results: **Accuracy:** We compared our results with those obtained using Ansys Icepak. The method involves two stages: the pre-compute stage which is off-line, in which we calculate the leakage aware Green's functions, and the compute stage, which is online, in which we convolve the Green's function with the power profile.

Pre-compute stage: We found that the maximum error in calculating the Green's functions using the Fourier transform was 0.2 °C in all cases. In terms of percentage, the maximum error was limited to 3%. Using Hankel transforms, the percentage errors were lower in some cases and higher in others. The average error was 5.5%. Since the Hankel transform is the 1D equivalent of a 2D Fourier transform for radially symmetric functions, the use of either of these two transforms should yield the same results. However we observe that the Hankel transform based approach has a higher error since there are sophisticated packages available for computing the Fourier transform, whereas the code for the Hankel transform has been implemented by us and thus there are more issues with precision. We found out that the process of calculating the Hankel transform and the inverse Hankel transform yields an error between 1-7%, with an average error of 4%. To correct for these numerical errors, we added a correction factor of 1.04. This reduced our average error to less than 3.5%.

Compute stage: We computed the thermal profiles for two test cases.

Test case 1: This is a simple test case in which each layer has a single power source other than the middle layer, which has two power sources. The power profile is described in Table III. The error in the total temperature profile was limited to 0.46 °C using Fourier transforms and 1.5 °C using Hankel transforms. The maximum temperature rise was 28.2 °C , resulting in a thermal estimation error of 5%.

Test case 2: We also tested our chip on a real floorplan consisting of one core layer (Alpha 21264) and 3 memory layers. The power values were taken from the 3D test case in HotSpot [7]. The power profile and the corresponding thermal



(a) Dynamic power dist., layer 1, test case 2 (b) Calculated thermal profile, layer 1 (c) Actual thermal profile, layer 1

Fig. 5: Power and thermal profiles for test case 2 for a chip with heat sink

TABLE III: Location and magnitude of dynamic power sources on the chip in test case 1

Grid point	Layer	Power (W)
(12,5)	Layer 1	1 W
(6,5)	Layer 2	3 W
(9,8)	Layer 2	1 W
(9,8)	Layer 3	2 W
(12,13)	Layer 4	4 W

profiles are shown in Figure 5. The maximum error here too was limited to 5%. We can see in Figures 5b and 5c that the calculated and actual thermal profiles match almost completely.

Speed:

3DSimF pre-compute stage: Computing the Fourier transform of all the 16 spreading functions and calculating the leakage aware Green's functions in the transform domain requires $0.6\ ms$. These functions are then stored to be used later in the online stage.

3DSimF Compute stage: The runtime of the online compute stage is $0.4\ ms$.

3DSimH pre-compute stage: To compute the Hankel transform of the Green's functions we need $0.96\ s$. The higher computation time needed to calculate the modified Green's functions using Hankel transforms is because we need to compute the results of Bessel functions. However, since this part is offline, and does not need to be computed again unless the parameters change, the higher simulation time is not a problem. Then, for the second sub-stage, it takes an additional $85\ \mu s$ to calculate the leakage aware functions in the domain of Hankel transforms. Here, we have an 82X speedup as compared to *3DSimF*. This is due to the fact that the size of the problem is reduced by an order of magnitude, and since we calculate 16 Green's functions, we have significant savings in time.

3DSimH Compute stage: The running time of the compute stage is 0.4 ms for *3DSimH* (almost the same as *3DSimF*). The results are summarized in Table IV.

To compute the same functions using Ansys Icepak, we require 2-3 hours, depending on the number of iterations needed for convergence. We also calculated the leakage converged temperature values for a similar configuration using the latest version of Hotspot [7], and found the execution time to be $0.22\ s$ (as compared to $1\ ms$ for $3DSim$). Using 3D-ICE, the execution time is $1.1\ s$ to compute the leakage converged thermal profile (for acceptable accuracy, a grid size of 32×32 had to be used in 3D-ICE). Therefore, our algorithm provides a $220 - 1100X$ speedup as compared to other state of the art thermal simulators for steady state analysis of 3-D chips. The accuracy is either similar or better in some cases (see Table IV).

TABLE IV: Speed and accuracy of popular simulators (steady state)

Simulator	Chip-Heatsink		Chip-Microchannel	
	Time	Error (°C)	Time	Error (%)
Ansys	2 *hours*	–	3.5 hours	–
Hotspot	0.22 s	1.4 [7]	–	–%
3D-ICE	1.1 s	1.6	1.67s	3.4%
$3DSimF$	0.001 s	1.5	0.0035s	4%

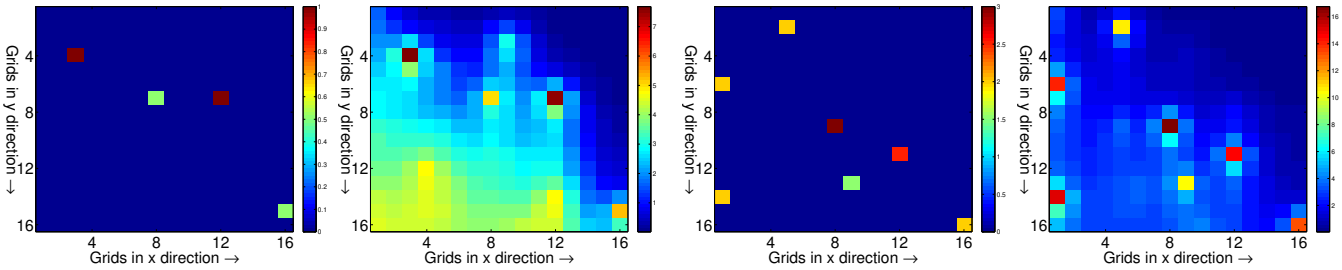
Test Case	Total power	Fluid vel.	Max. Temp. Rise (°C)	Error (°C)	Error (%)
Test case 1	9 W	1 m/s	16.74	0.04	0.2
Test case 2	31 W	1 m/s	16.68	0.14	0.8
Test case 3	64.3 W	1 m/s	12.54	0.26	2.1
Test case 4	64.3 W	0.75 m/s	13.73	0.25	1.8

TABLE V: Test cases used for evaluating chips with microchannels

2) Transient Results: For the transient temperature profile, the error obtained using our algorithm was limited to 6%. The execution time to obtain the leakage converged Green's functions depends on the number of points in time for which the simulation needs to be done. For 100 points between 0 and $0.05\ s$, the CPU execution time remains limited to $3.5\ s$. As discussed earlier, obtaining the leakage converged Green's functions using simulators such as Hotspot and 3D-ICE is prohibitively expensive since multiple iterations must be run for each point in time.

E. Results for 3D Chip with Microchannels

1) Steady State Results: **Green's functions:** We use the same approach using Ansys Icepak to obtain the *residual*



(a) Power distribution, layer 1 (b) Thermal profile, layer 1 (c) Power distribution, layer 4 (d) Thermal profile, layer 4

Fig. 6: Power and thermal profiles for test case 2 for layers 1 and 4 for a chip with microchannels

Green's function (source applied at the edges of every layer).

Pre-compute stage: Next, we calculate the leakage aware Green's functions for all layers using our algorithm. Our algorithm takes 3.5 ms to calculate all these Green's functions, with an error limited to 3%. We also calculate the leakage aware residual Green's functions, which takes an additional 3.5 ms. We do this for all leakage segments in the piecewise linear leakage model. We also obtain the damped Green's functions for each layer, which takes an additional 0.3 ms. Thus the total time spent in the pre-compute stage is 7.3 ms.

Compute stage: Using these Green's functions, we calculate the leakage aware full chip thermal profile for multiple test cases. The first two test cases are for the chip described in Section VI-A. In the third test case, we evaluate our algorithm on a real floorplan with one core layer and three memory layers. In the fourth test case, we vary the velocity of the fluid. A summary of the various test cases is shown in Table V. The simulation time of our algorithm is 1.2 ms when a single segment leakage model is used, and 3.5 ms when a 3-segment piecewise linear leakage model is used. To compute the same leakage aware thermal profile, Ansys Icepak requires 3-4 hours, while 3D-ICE takes 1.67 s. Thus our algorithm provides a **477X speedup over 3D-ICE** in the online part of thermal computation (150X speedup if the pre-compute time is included as well). Additionally our algorithm enables a tradeoff between simulation time and accuracy through the piecewise linear model.

Test Case 1: In this case we applied one power source to each layer of the 3D chip, except the layer closest to the ambient, where we apply two power sources. The total power applied to the chip is 9 W. The mean absolute error of our algorithm is limited to 0.04°C (maximum temperature rise of 16.7°C).

Test Case 2: In this case the total power applied to the chip is 31 W with multiple power sources in each layer. The mean absolute error of our algorithm is limited to 0.14°C for a maximum temperature rise of 16.68°C . The power and thermal profiles for each layer are shown in Figure 6.

Test Case 3: Here we have a core layer consisting of the processor Alpha21264 and 3 memory layers with uniform power dissipation. The total power dissipated by the core layer is 48.9 W, while the total power dissipated by the 3D chip stack is 64.3 W. Using our algorithm the mean absolute error is obtained to be 0.26°C for a maximum temperature rise of 12.54°C (Figure 7).

The temperature here is lower than test case 1 and 2, despite the total power being higher because a large fraction of the total power (48.9 W) is concentrated on the core layer, which is closest to the ambient, and has a better heat dissipation path. The power density for the remaining layers ($5W/cm^2$) is also lower than test case 1, where the power density is as high as $768W/cm^2$.

Test Case 4: Next we change the fluid velocity to 0.75 m/s, and keep the power profile the same as test case 3. The mean absolute error obtained in this case is 0.25°C for a maximum temperature rise of 13.73°C .

2) Transient Results: Pre-compute stage: We first compute the transient leakage aware Green's functions from the steady state Green's functions using the method outlined in Section III. The time taken to compute the transient thermal profile is 54 ms, for all the four layers (20 points in time: $2.7\ ms \times 20 = 54\ ms$). The maximum error is limited to 5%. The calculated transient leakage aware Green's function and the corresponding function obtained from Icepak are shown in Figure 9a.

Compute stage: Step Response: We need an additional 15 ms to compute the full chip thermal profile.

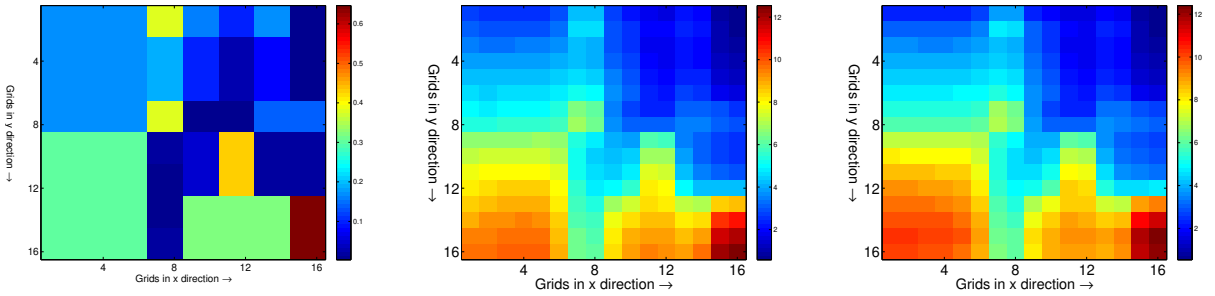
Test cases 1 and 2: We validate our algorithm for the floorplan of the Alpha21264 processor. The transient temperature profile using our algorithm and that obtained from Icepak for one of the grid locations are shown in Figure 9b. The error in this case was limited to 6%.

Next, we demonstrate the temperature profile corresponding to a power profile consisting of three power sources in a layer. The power profile and the corresponding temperature profile for layer 3 are shown in Figure 8.

To validate our results, we obtain the reference values from Icepak for 8-10 time instants only. To compute the leakage converged temperature values from Icepak, we had to set the mesh to the coarsest setting possible, otherwise our system ran out of memory. Even then, it took over 10 hours to obtain the full chip thermal profile from Icepak.

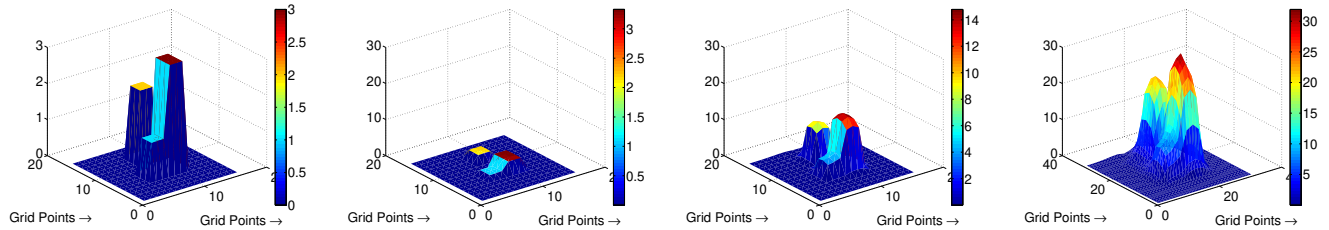
Compute stage: Time-varying Power Source: We compute the full chip transient thermal profile for test case 1, using the approach described in Section III-E. The calculated and the observed transient thermal response (using Icepak) at the center of the chip are shown in Figure 10. An average error of 1.9% is observed against the data from Ansys Icepak. The execution time for 80 time points is 300 ms.

Yan et al. [6] implement their algebraic multigrid preconditioned iterative solver in C++ and achieve a 9X to 139X



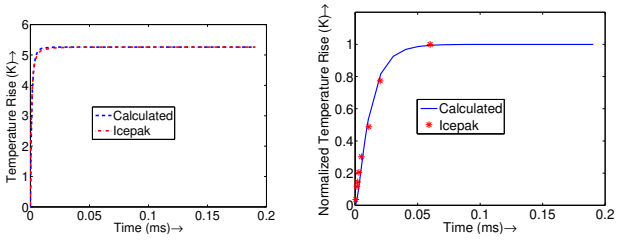
(a) Power distribution in core layer (b) Temperature map from Icepak (c) Calculated temperature map

Fig. 7: Power and temperature map for test case 3 for a chip with microchannels



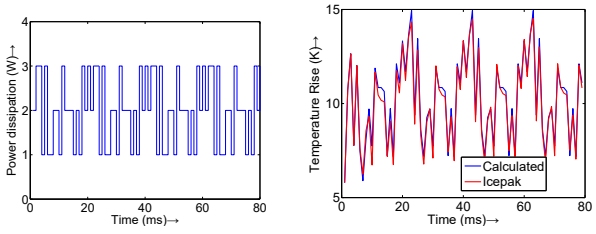
(a) Power distribution in layer 3 (b) Temperature rise at 0.5 ms (c) Temperature rise at 2.5 ms (d) Temperature rise at 7.5 ms

Fig. 8: Power and temperature map, transient



(a) Transient Green's function, layer 2 (b) Transient temperature profile, center grid, layer 4

Fig. 9: Transient temperature map for test case 1



(a) Transient power (b) Transient thermal profile

Fig. 10: Transient power and thermal distribution, layer 4

speedup over Hotspot. In comparison, our approach provides a 477X speedup in the steady state, when implemented in Matlab. If our algorithm is implemented in C++, we can have much larger speedups owing to our analytical approach.

F. Scalability Analysis of the Proposed Approach

To demonstrate the scalability of our algorithm to different grid sizes, we have carried multiple experiments using 3D-ICE. We have used test case 3 of chip with microchannels and simulated grid sizes of 4×4, 8×8, 16×16, 32×32 and 64×64. We report the execution time of our $3DSim$ algorithm and the 3DICE algorithm as well as the mean absolute error relative to 3D-ICE in Table VI. Our results show that we have a minimum speedup of 350X over 3D-ICE, and the gain improves as we move to finer grid sizes (upto 18000X at a 64×64 grid).

TABLE VI: Scalability to different grid sizes (steady state)

Grid Size	3DSim Time (s)		3DSim Error (K)	3DICE Time (s)	
	Pre-compute	Compute		Without leakage	With leakage
64×64	0.072 s	.015 s	0.22	67s	274s
32×32	.025 s	.0042 s	0.21	2.9s	22.6s
16×16	.0075 s	.0012 s	0.24	0.32s	1.6s
8×8	.0032 s	.0005 s	0.60	0.05s	0.35s
4×4	.002 s	.00034 s	1.96	0.03s	0.12s

G. Error Upon Using a Linear Leakage Model

Using a linear leakage model, we get a 5% error in temperature with the mean absolute error becoming 0.275K, from 0.25K in the piecewise linear leakage case. Whether this much error is acceptable or not will depend on the application.

VII. Related Work

A. Thermal Modeling of 2D Chips

For 2-D chips, a wide variety of techniques have been proposed for temperature simulation, which are fast as well as accurate [11], [18], [19]. Most temperature simulation techniques solve the Fourier's law of heat conduction using a finite element or finite difference based approach [18]. However, such methods are slow and the accuracy depends on the granularity of meshing.

The popular Hotspot [20] simulator uses the analogy between electrical and thermal circuits to construct an RC circuit, and then solves the resulting equations to get the thermal profile. To close the leakage-temperature loop, the authors suggest iterating multiple times till convergence.

Power Blurring is a Green's function based approach [2], where additional corrections are applied for edges and corners as well as the pyramid structure of the chip. The transient profile is modeled as well. But this method is not applicable to 3D chips, and models leakage iteratively.

Sarangi et al. [11] derive a leakage aware Green's function for 2D chips using a linear model of leakage. To convert the 2-D convolution to 1-D multiplication, they use the Hankel transform. At runtime, they convolve the total power profile with the Green's function to obtain leakage aware full chip temperature values. To obtain the transient temperature profile, they incorporate a thermal capacitance. This is the work closest to our area, but this method is for 2D chips, and owing to the multiple heating effects in both lateral and vertical directions, this method cannot be trivially extended to 3D chips. We consequently developed a novel technique to realistically approximate the Green's functions in a 3-D chip.

B. Modeling of Leakage Power Without Explicitly Iterating

Wang et al. [13] assume a linear leakage model, where the reference temperature in the Taylor expansion is updated when the node temperature differs from it by 10°C . They then reduce computations using a model order reduction technique. In comparison our transform based approach is much faster. Additionally, the authors have not modeled microchannels. Zhang et al. [21] use a piecewise linear leakage model in a finite element setup. They then use the exact Newton's method by partially linearizing the resulting equations. While their method is capable of modeling detailed geometries, they have not modeled microchannels and the method is still slow. Moreover, the finite element method is highly sensitive to the meshing, and altering the meshing even slightly may result in non-convergence. Yan et al. [6] assume a linear leakage model where the Taylor series expansion is carried out around a different reference temperature for each grid in the model. They then solve the corrected linearized equation using the algebraic multigrid method. However their solution is still iterative, and they consider the steady state only.

C. Thermal Modeling of 3D Chips

A limited number of techniques have been proposed for 3-D chips. MTA [22] uses the finite element method to compute the detailed thermal profile for complex 3D geometries. However, while being capable of accounting for detailed geometries, the tool is too slow for runtime applications. Additionally, such detailed structures are rarely of much use to the end user.

D. Thermal Modeling of Chips with Microchannels

Coskun et al. [23] model microchannels and TSVs by considering variable thermal resistivities for each grid cell in the mesh (created as a part of the finite difference based approaches). However, their method is slow and has limited accuracy. Mizunuma et al. [24] consider two *wake functions* for the temperature spreading effect due to microchannels. They do not model the transient thermal profile.

3D-ICE [3] is the most popular thermal simulator that models microchannels. Here, the effect of microchannel cooling is incorporated by adding a convective term for the heat exchange by the fluids. The additional term is modeled as a temperature controlled heat source, which translates to a voltage controlled current source in the equivalent RC circuit. We have compared our results against those obtained using 3D-ICE in Section VI, where we demonstrate that our method provides a 477X speedup over 3D-ICE. Qian et al. [25] account for the entrance effects at the inlet of microchannels. They also model TSVs by modifying the conductivity of the corresponding grid cells. Feng and Li [26] adapt this method to GPUs, but they do not model the transient problem. Liu et al. [27] solve this problem using a GPU-accelerated GMRES solver and demonstrate a significant speedup. However, all of these methods require specialized computational resources, whereas our algorithm requires a single core.

E. Machine Learning based Thermal Modeling

Juan et. al [28] propose a regression based framework to obtain the maximum temperature in a 3D IC in the presence of variation using a combination of the leakage power of each layer. Sridhar et al. [29] use a single layer neural network to predict future temperatures using the current temperature and power values as input for a 3D IC with microchannels. They implement the online part of their technique on a GPU. However they do not model leakage power. Other methods use feature selection over a large set of features to predict future temperatures [30], [31] Sadiqbatcha et. al [32] use an LSTM based model to augment the thermal sensor readings, followed by DCT for compression. Wang et. al [33] use a variant of RNN, echo state networks (ESN) to learn the relationship between power and temperature, while capturing the non-linear dependence of leakage power on temperature and avoiding the long term dependencies problem. However, they do not model 3D chips.

A major limitation of all machine learning based methods is that they are extremely sensitive to the training input. Additionally the large amount of training data needed by these algorithms can not always be obtained, especially considering leakage power. Getting the current temperature or performance counter readings quickly at runtime is challenging too, since there are several preprocessing steps involved before this data can be used. Also, such measurements are prone to noise and measurement delays. Thus our focus has been on *analytical* thermal models in this work.

VIII. Conclusion

In this work, we propose a fast analytical method to model the temperature profile of a 3D chip, which includes the effects of leakage and microchannels. We incorporate a piecewise linear leakage model for improved accuracy. We compared our results with a commercial CFD simulator as well as state of the art thermal modeling tools. Our approach yields a significant speedup with an accuracy comparable to other state of the art tools. The high speedup is because we adapt a Green's based

function approach, and obtain a closed form solution, which seamlessly incorporates the effect of leakage; this eliminates the need to perform multiple iterations for leakage-temperature convergence and essentially converts a 3-D problem to a primarily 1-D problem.

ACKNOWLEDGMENT

This work has been partially supported by the Semiconductor Research Corporation (SRC) under Grant No. 2017-CT2735. The first author is supported by Visvesvaraya PhD Scheme, MeitY, Govt. of India MEITY-PHD-701. We would like to thank Shashank Varshney for his valuable contributions.

REFERENCES

[1] T. Brunschwiler, B. Michel, H. Rothuizen, U. Kloter, B. Wunderle, H. Oppermann, and H. Reichl, "Interlayer cooling potential in vertically integrated packages," *Microsystem Technologies*, vol. 15, no. 1, pp. 57–74, 2009.

[2] A. Ziabari, J.-H. Park, E. K. Ardestani, J. Renau, S.-M. Kang, and A. Shakouri, "Power blurring: Fast static and transient thermal analysis method for packaged integrated circuits and power devices," *VLSI Systems, IEEE Transactions on*, vol. 22, no. 11, pp. 2366–2379, 2014.

[3] A. Sridhar, A. Vincenzi, M. Ruggiero, T. Brunschwiler, and D. Atienza, "3D-ICE: Fast compact transient thermal modeling for 3D ICs with inter-tier liquid cooling," in *ICCAD, 2010*.

[4] T.-Y. Wang, Y.-M. Lee, and C. C.-P. Chen, "3D thermal-ADI: an efficient chip-level transient thermal simulator," in *ISPD, 2003*.

[5] W. Huang, E. Humenay, K. Skadron, and M. R. Stan, "The need for a full-chip and package thermal model for thermally optimized ic designs," in *ISLPED'05*. IEEE, 2005, pp. 245–250.

[6] C. Yan, H. Zhu, D. Zhou, and X. Zeng, "An efficient leakage-aware thermal simulation approach for 3d-ics using corrected linearized model and algebraic multigrid," in *DATE*. IEEE, 2017, pp. 1207–1212.

[7] R. Zhang, M. R. Stan, and K. Skadron, "Hotspot 6.0: Validation, acceleration and extension," University of Virginia, Tech. Rep., 2015.

[8] W. Liu, K. Cao, X. Jin, and C. Hu, "BSIM 4.0.0 technical notes," University of California, Berkeley, Tech. Rep. UCB/ERL M00/39, 2000.

[9] H. Sultan, A. Chauhan, and S. R. Sarangi, "A survey of chip-level thermal simulators," *ACM Computing Surveys*, vol. 52, no. 2, p. 42, 2019.

[10] Y. Liu, R. P. Dick, L. Shang, and H. Yang, "Accurate temperature-dependent integrated circuit leakage power estimation is easy," in *DATE, 2007*.

[11] S. R. Sarangi, G. Ananthanarayanan, and M. Balakrishnan, "Lightsim: A leakage aware ultrafast temperature simulator," in *ASPDAC, 2014*.

[12] H. Sultan and S. R. Sarangi, "A fast leakage aware thermal simulator for 3d chips," in *DATE*. IEEE, 2017, pp. 1733–1738.

[13] H. Wang, J. Wan, S. X.-D. Tan, C. Zhang, H. Tang, Y. Yuan, K. Huang, and Z. Zhang, "A fast leakage-aware full-chip transient thermal estimation method," *IEEE Transactions on Computers*, vol. 67, no. 5, pp. 617–630, 2018.

[14] P. Zhou, Y. Ma, Z. Li, R. P. Dick, L. Shang, H. Zhou, X. Hong, and Q. Zhou, "3D-STAF: scalable temperature and leakage aware floorplanning for three-dimensional integrated circuits," in *ICCAD, 2007*.

[15] I. Wolfram Research, "Mathematica," Illinios, 2012.

[16] H. F. Johnson, "An improved method for computing a discrete hankel transform," *Computer physics communications*, vol. 43, no. 2, pp. 181–202, 1987.

[17] Y. Yang, Z. Gu, C. Zhu, R. P. Dick, and L. Shang, "Isac: Integrated space-and-time-adaptive chip-package thermal analysis," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 26, no. 1, pp. 86–99, 2006.

[18] W. Huang, K. Skadron, S. Gurumurthi, R. J. Ribando, and M. R. Stan, "Differentiating the roles of IR measurement and simulation for power and temperature-aware design," in *ISPASS, 2009*.

[19] J.-H. Park, S. Shin, J. Christofferson, A. Shakouri, and S.-M. Kang, "Experimental validation of the power blurring method," in *SEMI-THERM*. IEEE, 2010, pp. 240–244.

[20] W. Huang, S. Ghosh, S. Velusamy, K. Sankaranarayanan, K. Skadron, and M. R. Stan, "Hotspot: A compact thermal modeling methodology for early-stage VLSI design," *VLSI Systems, IEEE Transactions on*, vol. 14, no. 5, pp. 501–513, 2006.

[21] C. Zhang, M. Mihajlović, and V. F. Pavlidis, "Adaptive transient leakage-aware linearised model for thermal analysis of 3-d ics," in *DATE*. IEEE, 2019, pp. 268–271.

[22] S. Ladenheim, Y.-C. Chen, M. Mihajlović, and V. F. Pavlidis, "The mta: An advanced and versatile thermal simulator for integrated systems," *IEEE TCAD*, vol. 37, no. 12, pp. 3123–3136, 2018.

[23] A. K. Coskun, J. L. Ayala, D. Atienza, and T. S. Rosing, "Modeling and dynamic management of 3d multicore systems with liquid cooling," in *VLSI-SoC*. IEEE, 2009, pp. 35–40.

[24] H. Mizunuma, C.-L. Yang, and Y.-C. Lu, "Thermal modeling for 3d-ics with integrated microchannel cooling," in *ICCAD*. ACM, 2009, pp. 256–263.

[25] H. Qian, H. Liang, C.-H. Chang, W. Zhang, and H. Yu, "Thermal simulator of 3d-ic with modeling of anisotropic tsv conductance and microchannel entrance effects," in *ASP-DAC 2013*.

[26] Z. Feng and P. Li, "Fast thermal analysis on gpu for 3d ics with integrated microchannel cooling," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 21, no. 8, pp. 1526–1539, 2013.

[27] X.-X. Liu, Z. Liu, S. X.-D. Tan, and J. Gordon, "Full-chip thermal analysis of 3d ics with liquid cooling by gpu-accelerated gmres method," in *ISQED*. IEEE, 2012, pp. 123–128.

[28] D.-C. Juan, S. Garg, and D. Marculescu, "Statistical thermal evaluation and mitigation techniques for 3d chip-multiprocessors in the presence of process variations," in *DATE 2011*.

[29] A. Sridhar, A. Vincenzi, M. Ruggiero, and D. Atienza, "Neural network-based thermal simulation of integrated circuits on gpus," *IEEE TCAD*, vol. 31, no. 1, pp. 23–36, 2011.

[30] J. M. N. Abad and A. Soleimani, "Novel feature selection algorithm for thermal prediction model," *IEEE TVLSI*, vol. 26, no. 10, pp. 1831–1844, 2018.

[31] K. Zhang, A. Guliani, S. Ogrenci-Memik, G. Memik, K. Yoshii, R. Sankaran, and P. Beckman, "Machine learning-based temperature prediction for runtime thermal management across system components," *IEEE Transactions on parallel and distributed systems*, vol. 29, no. 2, pp. 405–419, 2017.

[32] S. Sadiqbatcha, Y. Zhao, J. Zhang, H. Amrouch, J. Henkel, and S. X.-D. Tan, "Machine learning based online full-chip heatmap estimation," in *2020 25th Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE, 2020, pp. 229–234.

[33] H. Wang, X. Guo, S. X.-D. Tan, C. Zhang, H. Tang, and Y. Yuan, "Leakage-aware predictive thermal management for multi-core systems using echo state network," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2019.

Hameedah Sultan Hameedah Sultan is currently working towards her Ph.D. degree in the School of Information Technology, Indian Institute of Technology Delhi. She has done her Masters in VLSI Design Tools and Technology, IIT Delhi. Her research interests include architectural-level thermal and noise modeling.

Smruti R. Sarangi Prof. Smruti Ranjan Sarangi is an Associate Professor in the Computer Science and Engineering Department at IIT Delhi with a joint appointment in the Department of Electrical Engineering. He primarily works in parallel and distributed architectures and systems. His research areas cover multicore processors, cyber-security, emerging technologies, networks on chip, operating systems for parallel computers, and parallel algorithms. Dr. Sarangi obtained his Ph.D in computer architecture from the University of Illinois at Urbana Champaign(UIUC), USA in 2006, and a B.Tech in computer science from IIT Kharagpur in 2002. He has filed five US patents, three Indian patents, and has published 87 papers in reputed international conferences and journals. He is the author of the popular undergraduate textbook on computer architecture titled, "Computer Organisation and Architecture", published by McGrawHill. He is a member of the IEEE and ACM.