A Survey of Chip Level Thermal Simulators

HAMEEDAH SULTAN, School of Information Technology, Indian Institute of Technology, India
ANJALI CHAUHAN, School of Information Technology, Indian Institute of Technology, India
SMRUTI R. SARANGI, Computer Science and Engg., Indian Institute of Technology, India

Abstract
Thermal modeling and simulation have become imperative in recent years owing to the increased power density of high performance microprocessors. Temperature is a first order design criteria, and hence special consideration has to be given to it in every stage of the design process. If not properly accounted for, temperature can have disastrous effects on the performance of the chip, often leading to failure. In order to streamline research efforts, there is a strong need for a comprehensive survey of the techniques and tools available for thermal simulation. This will help new researchers entering the field to quickly familiarize themselves with the state of the art, and enable existing researchers to further improve upon their proposed techniques. In this paper we present a survey of the package level thermal simulation techniques developed over the last two decades.

CCS Concepts: • **Hardware → Temperature simulation and estimation**; *3D integrated circuits*; *Chip-level power issues*; Modeling and parameter extraction.

Additional Key Words and Phrases: Thermal simulation, Finite Element, Finite Difference, Green's function, Machine Learning, 2D chips, 3D chips, Microchannels, Leakage, Fourier equation, Boltzmann Equation.

ACM Reference Format:
Hameedah Sultan, Anjali Chauhan, and Smruti R. Sarangi. 2019. A Survey of Chip Level Thermal Simulators . *ACM Trans. Graph.* xx, x, Article 000 (2019), 35 pages. https://doi.org/xx.xxxx/xxxxxxx.xxxxxxx

1 INTRODUCTION

Thermal challenges have become one of the major limiting factors in the high performance processor industry. Non-uniform power consumption in functional units creates a non-uniform temperature distribution, which in turn leads to undesirable thermal hot spots. An unmanageably high temperature accelerates failure mechanisms [8] such as electromigration and dielectric breakdown. These issues are aggravated with the scaling of the devices. High temperature affects the mobility of the carriers, causes negative-bias temperature instability (NBTI) and hot carrier injection (HCI), which degrades the performance and reliability of the chip [5, 49]. Due to shorter interconnects, the resistivity increases with temperature leading to large IR drops and longer RC delays [2, 69]. This results in performance loss, and further complicates timing and noise analysis. Moreover, leakage power increases superlinearly with temperature leading to thermal runaways and IC breakdown.

With the miniaturization of technology, thermal issues are more adverse in three-dimensional (3D) chips because of the higher power density, and limitations of air cooling [40, 66]. 3D chips

Authors' addresses: Hameedah Sultan, School of Information Technology, Indian Institute of Technology, Delhi, India, hameedah.sultan@gmail.com; Anjali Chauhan, School of Information Technology, Indian Institute of Technology, Delhi, India; Smruti R. Sarangi, Computer Science and Engg., Indian Institute of Technology , Delhi, India, srsarangi@cse.iitd.ac.in.

0730-0301/2019/0-ART000 $15.00
https://doi.org/xx.xxxx/xxxxxxx.xxxxxxx

were originally proposed to reduce the latency between the processor, memory and other logic elements and increase the instruction throughput [18, 75]. The advantage of 3D packaging is that the total area required to implement the same functionality is less than an equivalent 2D chip [66]. On the down side, the complexity of the 3D chip design concentrates heat sources; this traps the heat and causes severe thermal issues. In 2D chips, the spreader and heat sink serve to limit the temperature rise whereas in 3D chips, the dissipation of heat through the spreader and sink is not very effective because of the presence of multiple layers. The 3D stacking of active layers increases the thermal resistance from the heated layer to the heat sink; heat cannot be effectively removed by air-cooling, and the consequent temperature rise becomes a limiting factor.

An accurate estimate of temperature requires an accurate estimate of leakage power, since leakage power strongly depends on temperature. As the leakage power increases, the on-chip temperature increases, resulting in a positive feedback effect [79].

All these factors make it imperative to model temperature effects considering leakage power in modern processors, before the detailed layout is available. A method to estimate temperature in the early stages of design can help in thermal characterization of the chip, along with temperature and power optimization, floorplanning and cell placement [16, 27, 33, 63, 99, 100].

The need for thermal simulators exists in the entire computer architecture and systems research communities. Several researchers have worked towards the development of thermal simulators and optimized them for different design criteria. In order to carry out several design optimizations, a quick method to predict the thermal profile for any given floorplan and power profile is needed. For thermal aware floorplanning, some methods such as simulated annealing need to perform millions of thermal simulations to determine the optimal floorplan [14, 34]. In such cases, the speed of the thermal modeling approach is the most important factor, and accuracy can be compromised to some extent. A very small speed-up can save large amounts of time when repeated millions of times. Also, the level of accuracy needed at the architectural level is not very high (more on this in Section 8).

Often researchers use over-simplistic methods for thermal simulation, either because thermal simulators that cater to their requirements do not exist, or are not known to the researchers. We believe that a survey of the existing architectural thermal simulation techniques will help researchers identify the simulator best suited to their needs. It will also help researchers working on thermal simulation techniques determine the shortcomings of the current techniques, leading to further improvements in this area.

Let us quickly discuss some previous efforts in this direction. A summary of the architectural techniques for thermal simulation was presented in [82]. A brief overview of the thermal issues and techniques to deal with them was described in [62]. A more extensive survey was done by Zhan et al. [93]. However all of these surveys were done before 2007, when a very limited number of techniques were available for full chip thermal simulation. Thus, we believe that a detailed survey of the techniques has long been in order, and furthermore there is a dire need to cover all the techniques proposed in the last 15 years.

2 BACKGROUND

The following section discusses the structure of a chip, the heat transfer mechanism, and an overview of the various methods for solving the heat equation. The notations and symbols used in this paper are listed in Table 1. The organization of the paper is summarized in Figure 1.

2.1 Structure of a Chip in an IC

2D chips consist of an active silicon layer called the *die* attached to a ceramic carrier called the *substrate* using an array of solder balls. The remaining space between the soldered balls is filled

with an underfill material called *epoxy*. On the back side, the die is attached to the heat spreader (alloy of nickel and copper) via a thermal interface material. The heat spreader in turn is connected to a heat sink for effective air cooling. The whole package is then mounted on the PCB using *through hole* or *surface mount* technologies.

A 3D chip contains more than one active silicon layers separated by a thermal interface material. It must be noted that just like a 3D chip, a 2D chip package too consists of several layers of different materials. In earlier papers, this assembly was often referred to as a 3D chip. However, in this paper, we consider 3D chip to imply a chip containing more than one active (heat dissipating) layers of silicon. A typical 3D chip is shown in Figure 2.

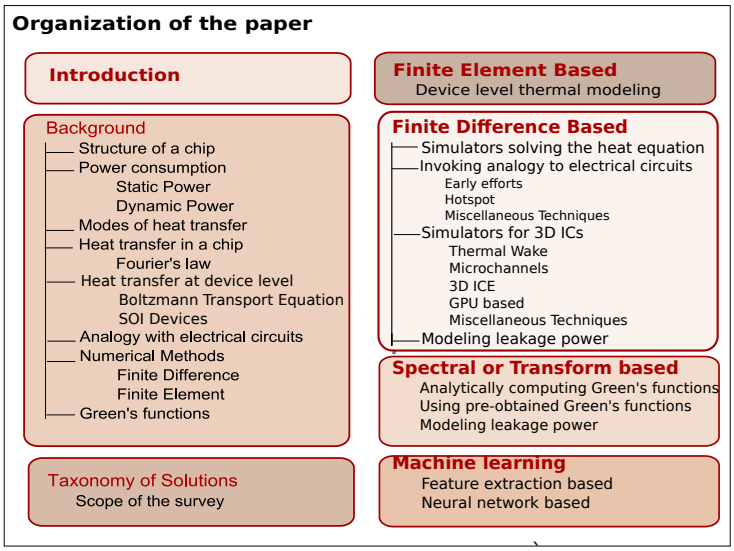

Fig. 1. Organization of the paper

Symbol	Meaning
T	Temperature
v_T	Thermal voltage
V	Voltage or potential
C	Capacitance
t	Time
Q	Heat
κ	Thermal conductivity
A	Area
h	Heat transfer coefficient
$\dot{q}$	Heat generation rate
C_v	Volumetric specific heat
ρ	Density
R	Thermal resistance
G	Green's function
P	Power

Table 1. Notations used in this paper

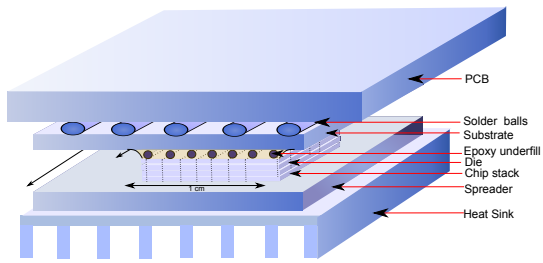

Fig. 2. A typical 3D chip

The silicon die contains the active power dissipating components. The heat spreader has two functions: facilitate effective vertical heat transfer, and homogenize the temperature profile through lateral heat transfer. The heat sink is typically a multi-fin heat exchanger that exchanges heat with the ambient using either normal or forced convection. We use multiple fins (see Figure 2) to increase the surface area such that more heat can be effectively dissipated. In some designs such as modern desktops, there is a small fan on top of the heat sink. It blows air at a high velocity over the heat sink, and this increases the rate of heat dissipation. Fans can additionally be present at the

end of the mother board if the cooling requirements are higher. This configuration is often used in high end servers.

2.2 Power Consumption in a Chip

The power dissipated by a chip can broadly be classified into two types: *static* power and *dynamic power*.

2.2.1 Static Power Dissipation. Static or leakage power is the power dissipated when the processor is not running any workload. The most dominant source of leakage current is the *subthreshold leakage*. It occurs when a small amount of current flows even when the *gate-to-source voltage*, V_{GS} in a MOSFET is less than the threshold voltage, V_{th}. It is a function of temperature and can be described by the simplified BSIM 4 [50] model, as given in Equation 1.

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

where υ_T is the thermal voltage (kT/q), V_{th} is the threshold voltage, V_{ds} is the drain-source voltage,V_{gs} is the gate-source voltage, V_{off} is the offset voltage in the sub-threshold region and η is a constant.

The other sources of leakage power do not have such a strong temperature dependence and can be modeled by a constant.

2.2.2 Dynamic Power Dissipation. The dynamic power dissipation is dependent on the switching activity and is independent of temperature. It is given by Equation 2.

$$P_{dyn} = \alpha C V^2 f, \tag{2}$$

where, V is the supply voltage, f is the operating frequency and α represents the switching activity in the circuit.

2.3 Modes of Heat Transfer

At the nano-scale level thermal energy is transferred via the kinetic energy of the molecules. As temperature increases, the molecules vibrate more vigorously, and they transfer this energy to neighboring molecules with low kinetic energy. Once thermal equilibrium is established, the heat transfer stops. There are three primary means of heat transfer: conduction, convection and radiation.

Conduction: Conduction is the "transfer of energy from the more energetic to the less energetic particles of a substance due to interactions between the particles" (quoted directly from [6]). Conduction is usually the primary means of heat transfer within solids. Heat transfer by conduction depends on the material of the conducting medium, temperature difference across conducting faces, cross sectional area of the conducting faces, and the distance between them. For a planar surface, the amount of heat transfer by conduction is given by:

$$\frac{Q}{t} = \frac{\kappa A(T_a - T_b)}{l}, \tag{3}$$

where κ is the thermal conductivity, Q is the amount of heat transferred in time t, A is the cross sectional area of the material, l is the length of the conductor, T_a is the temperature of the hotter end, and T_b is the temperature of the colder end. Thermal conductivity is the inverse of thermal resistivity, and is a material constant.

A more general equation for heat transfer by conduction is given in Section 2.4.1.

Convection: Convection occurs when a fluid (liquid or gas) is heated. The molecules gain energy and expand, becoming lighter and move away from the source of the heat. They carry their thermal

energy along with them. The cold fluid replaces the hot fluid, resulting in convection currents which transport energy. Convection is usually the primary means of heat transfer within liquids and gases. Convection can be *natural* or *forced*. Forced convection occurs when a liquid or gas is forced to flow over a surface by an external source such as a fan. Heat is removed from heat sinks usually using forced convection. The rate of convection can be calculated by Equation 4.

$$\frac{Q}{t} = h_c A(T_s - T_f), \tag{4}$$

where h_c is the convective heat transfer coefficient, T_s is the surface temperature, and T_f is the temperature of the bulk fluid.

Radiation: Heat transfer through radiation occurs when a body emits thermal energy in the form of electromagnetic radiations because of thermal agitation of its charged particles. All materials emit thermal radiation at any temperature greater than absolute zero; the wavelength and frequency of the emitted waves depend on the temperature of the body.

The power radiated by a body, P, is given by the Stefan-Boltzmann law:

$$P = A\epsilon\sigma T^4, \tag{5}$$

where σ is the Stefan Boltzmann constant, A is the area of the radiating surface, and ϵ is the emissivity. Emissivity "provides a measure of how efficiently a surface emits energy relative to a blackbody" (quoted directly from [6])

2.4 Heat Transfer in a Chip

In a chip, heat is transferred through two main paths. 1) The primary path is through the chip, upwards to the heat sink, via conduction. Heat is removed from the heat sink by convection. 2) The secondary path is in the reverse direction, from the die to the PCB via the ball grid array. To avoid heating of the PCB, we want this component to be as small as possible.

2.4.1 Fourier's Law of Heat Conduction. The Fourier's law of heat conduction governs the heat removal rate because of conduction. It is based on the principle of conservation of energy. It states that the heat removal rate is proportional to the temperature gradient. The general form of the Fourier's law is given by Equation 6.

$$\nabla.[\kappa\nabla T] + \dot{q} = \rho C_v \frac{\partial T}{\partial t}, \tag{6}$$

where the notations are listed in Table 1. In terms of matrices, it can be described as:

$$GT(t) + C\dot{T}(t) = P(t), \tag{7}$$

where G is the thermal conductance matrix, C is the thermal capacitance matrix and P and T are the power and temperature matrices respectively.

Under steady state conditions, for a homogeneous material, it is reduced to:

$$-\kappa\nabla^2 T = \dot{q} \tag{8}$$

For the one dimensional case, it can be re-written as:

$$-\kappa\frac{\partial^2 T}{\partial x^2} = \dot{q} \tag{9}$$

2.5 Heat transfer at the Device Level

At the level of transistors, quantum effects come into play because of additional energy transfer in the form of lattice vibrations, also called as phonons. Modern transistors have features sizes comparable to the size of a few atoms. These atoms are arranged in the form of a lattice. The vibration of atoms in this lattice leads to the transfer of energy, usually in the form of heat. This energy transfer is quantized and represented by the transfer of phonons. The classic Fourier equation fails to take into account the nanoscale heat transfer effects. This happens when either the device dimensions become comparable to the phonon mean free path ($\sim 300\ nm$ in silicon at room temperature), or the time scale approaches the relaxation time of energy carriers. There are three methods to take into account the effects of these phonons: *molecular dynamics method*, the *Boltzmann transport equation* and the *ballistic-diffusive method*. The molecular dynamics method, though quite accurate is very computationally expensive. The Boltzmann transport equation is valid when the wave nature of the energy carriers can be neglected. This happens when the distances in consideration are larger than the phonon wavelength ($\sim 0.1\ nm$), which is the case in electronic devices. Hence all device level thermal estimation techniques that generate the thermal profile for a full chip solve the Boltzmann transport equation in some form. However, the transient form of the basic Boltzmann equation is very difficult to solve. Hence researchers have used several approximations to solve it. The ballistic-diffusive method is one such approximation.

2.5.1 The Boltzmann Transport Equation. The Boltzmann transport equation (BTE) under the relaxation time approximation ($\frac{\partial f}{\partial t}|_{collision} = -\frac{f-f_0}{\tau(\omega)}$) is given by Equation 10:

$$\frac{\partial f}{\partial t} + \mathbf{v}.\nabla f - q_{vol} = -\frac{f - f_0}{\tau(\omega)},, \tag{10}$$

where f is the phonon energy distribution function, f_0 is the equilibrium energy density, $\mathbf{v}$ is the group velocity and q_{vol} is the volumetric heat generation.

The Gray-BTE is obtained by assuming that all phonons have the same mode (ω), group velocity ($\mathbf{v}$) and relaxation time (τ_{eff}). A solution of the Gray-BTE gives the nanoscale level temperature profile.

2.5.2 SOI Devices. Silicon on Insulator (SOI) devices have a buried oxide layer on top of which the device is fabricated. The thermal conductivity of silicon in SOI devices is modified by the presence of this oxide layer. Because of increased scattering in thin film silicon (thickness of silicon layer less than the mean free path of phonons), the lateral thermal conductivity can be significantly lower than that of bulk silicon. The change in conductivity is dependent on the thickness of the oxide layer as well as the temperature. This problem is exacerbated in FinFETs because of the small fin dimensions. Vasileska et al. [83] have followed the approach of Sondheimer [74] and used the following equation to obtain the modified conductivity of a thin film silicon with thickness a along the z direction:

$$\kappa(z) = \kappa_0(T) \int_0^{\pi/2} sin^3(\theta) \left\{ 1 - exp\left(\frac{-a}{2\lambda(T)cos\theta}\right) cosh\left(\frac{a-2z}{2\lambda(T)cos\theta}\right) \right\} d\theta, \tag{11}$$

where $\lambda(T)$ is the mean free path of phonons, a is the thickness of the silicon film and $\kappa_0(T)$ is the bulk thermal conductivity of silicon as a function of temperature, T.

They then model the mean free path of phonons as a function of temperature, and use these as inputs to an electrothermal device simulator based on Monte Carlo simulations coupled with BTE. The authors report the maximum temperature using bulk silicon conductivity values to be 400K, while those using the modified conductivity are over 500K for a 25 nm channel-length transistor,

Fig. 3. An electrical circuit

with an approximately 6-15% decrease in source-drain current. These results show the importance of modeling the quantum effects for SOI devices.

Xu et al. [54] model an SOI FinFET by dividing it into five regions, and treating each region separately. For each region, the authors modify the heat equation and apply appropriate boundary conditions. Finally the solutions from all the regions are coupled. This method is extended to perform transient thermal simulation as well.

There are a large number of works that deal with accurate thermal modeling of FinFETs and SOI CMOS. However, we were not able to find a single paper on the impact of SOI devices on the chip level thermal profile. We have covered only two representative thermal modeling techniques for SOI devices, since our focus is on architectural level simulators.

2.6 Analogy with Electrical Circuits

The analogy between electrical and thermal circuits is often used to model heat flow, with temperature being represented as voltage and heat flow represented as current. This is because the differential equations governing current and voltage are the same as the ones governing heat flow and temperature difference. The Poisson equation in electrostatics is given by Equation 12.

$$\nabla^2 V = \frac{-\rho}{\epsilon_0}, \tag{12}$$

where V is the electric potential, ρ is the charge density and ϵ_0 is a medium dependent constant, known as the *permittivity* of the medium. This equation is similar to Equation 8, where temperature is analogous to electric potential, conductivity is analogous to permittivity, and the rate of heat flow is analogous to charge distribution.

In an electrical circuit, a potential difference between two nodes is needed for the flow of current. Similarly, in a thermal circuit, a temperature difference is needed for heat flow. In Figure 3, the current is given by:

$$I = \frac{V_1 - V_2}{R} \tag{13}$$

Similarly, in a thermal circuit, the heat flow is given by:

$$Q = \frac{T_1 - T_2}{R}, \tag{14}$$

where $T_1 - T_2$ is the temperature difference between the two end points of a 1-dimensional structure, Q is the heat flow, and R is the thermal resistance. *Thermal resistance* is the opposition offered to the flow of heat, and is similar in properties to electrical resistance. When conduction is the dominant means of heat transfer, the thermal conductance (K) is given by:

$$K = \frac{\kappa A}{L} \tag{15}$$

When convection is the primary mode of heat transfer, the thermal resistance is:

$$R = \frac{1}{hA}, \tag{16}$$

where h is the convective heat transfer coefficient. In thermal systems, the rate of change of temperature with heat flow is given by:

$$Q = C\frac{dT}{dt}, \tag{17}$$

where $\frac{dT}{dt}$ is the rate of change of temperature, and C is the thermal capacitance.

Similar to thermal resistance, thermal capacitance captures the transient effects of heat transfer. The thermal capacitance is proportional to the length and the area of cross section of heat transfer:

$$C = \frac{cA}{l}, \tag{18}$$

where c is the thermal capacitance per unit volume.

2.7 Numerical Methods

The heat transfer equation is typically solved numerically using finite difference and finite element methods. Let us elaborate.

2.7.1 Finite Difference Method (FDM). For the sake of simplicity, we discuss the method for a single dimension. The Taylor series expansion of a function, $f(x)$, is given by:

$$f(x+\Delta x) = f(x) + \frac{\partial f}{\partial x}\Delta x + \frac{\partial^2 f}{\partial x^2}\frac{(\Delta x)^2}{2} + \cdots \frac{\partial^n f}{\partial x^n}\frac{(\Delta x)^n}{n!} + \cdots \tag{19}$$

We discretize the integration space in the x axis, to generate a sequence of small segments. The temperature for segment $i+1$ can be expressed in terms of the temperature for segment i using Taylor series expansion. Let us denote the temperature for segment i as T_i. Thus we have:

$$T_{i+1} = T_i + \left(\frac{\partial T}{\partial x}\right)_i \Delta x + \left(\frac{\partial^2 T}{\partial x^2}\right)_i \frac{(\Delta x)^2}{2} + \left(\frac{\partial^3 T}{\partial x^3}\right)_i \frac{(\Delta x)^3}{6} + \cdots \tag{20}$$

Further simplifying, we arrive at:

$$\left(\frac{\partial T}{\partial x}\right)_i = \frac{T_{i+1} - T_i}{\Delta x} - \left(\frac{\partial^2 T}{\partial x^2}\right)_i \frac{(\Delta x)}{2} - \left(\frac{\partial^3 T}{\partial x^3}\right)_i \frac{(\Delta x)^2}{6} + \cdots \tag{21}$$

This is the forward difference, since it is based on the difference in the values at segment (i) and $(i+1)$. A similar relation can be obtained using the backward difference of $T(i-1)$ and $T(i)$:

$$\left(\frac{\partial T}{\partial x}\right)_i = \frac{T_i - T_{i-1}}{\Delta x} + \cdots \tag{22}$$

Similarly, a central difference can be obtained by subtracting $T(i-1)$ and $T(i+1)$:

$$\left(\frac{\partial T}{\partial x}\right)_i = \frac{T_{i+1} - T_{i-1}}{2\Delta x} + \cdots \tag{23}$$

On these lines, for a given line segment we can create a set of equations. These equations are like recurrence relations, where they describe the temperature of segment $i+1$ based on the temperature of the k previous segments. These equations use discrete variables, and can be solved using standard techniques in linear algebra, after applying appropriate boundary conditions. This method will thus yield the temperature profile of the 1D structure. We can extend also this method on similar lines to compute the temperature in complex 2D and 3D geometries. We shall simply have more equations. However, they can be solved very easily using algebraic techniques.

2.7.2 Finite Element Method (FEM). The Finite Element method is another technique used to compute the numerical solution to a partial differential equation.

Basic Principle: The domain of interest is divided into several smaller subsections called *elements*. These elements share boundaries or surfaces and are interconnected at joints known as *nodes*. The complete assembly of elements is called a *mesh*. The field quantity is described over the complete domain by a set of partial differential equations that are difficult to solve analytically. Hence an approximate function is chosen for each element that approximates the original problem for that region. The governing equations for each element are calculated and then assembled to yield equations of the type $[k][T] = [Q]$, where k is the global stiffness matrix, T is the temperature matrix and Q is the heat flux matrix. The stiffness matrix is formulated based on the material properties of the system. The boundary conditions are then applied and the resulting set of equations are solved to obtain temperature values for all elements. The accuracy of this method depends on the number of subregions chosen for analysis.

Galerkin Method: The most common way to solve a problem using the finite element method is using a technique called the *variational formulation*. Let us solve the following differential equation to illustrate the Galerkin method (example of a variational method):

$$\dot{T}(x) = \lambda T(x) \tag{24}$$

First we multiply Equation 24 with a test function $v(x)$ belonging to a function space V. Then we integrate over the interval 0 to N, where N represents the range of x [4].

$$\begin{aligned} &\int_0^N \dot{T}(x)v(x)dx = \lambda \int_0^N T(x)v(x)dx, \text{ for all } v \in V \\ &\text{or} \\ &\int_0^N (\dot{T}(x) - \lambda T(x))v(x)dx = 0, \text{ for all } v \in V \end{aligned} \tag{25}$$

Assuming that the solution T belongs to the same function space V as the test function, we perform an integration. This is called the variational formulation or the weak formulation, since it is obtained by relaxing the original equation. The resulting equations are discretized to obtain a set of numerical equations. The goal is to obtain an approximate solution T_h. The approximate solution can be expressed as a sum of basis functions, ϕ_i, belonging to the reduced subspace of functions:

$$T_h(x) = \sum_i T_i \phi_i(x) \tag{26}$$

Let us assume the reduced subspace is the set of all polynomial equations of order less than or equal to p. Hence,

$$\begin{aligned} T_h(x) &= \sum_{i=0}^{p} T_i x^i = T_0 + T_1 x + T_2 x^2 + T_3 x^3 \ldots + T_p x^p \\ \dot{T_h}(x) &= \sum_{i=1}^{p} i T_i x^{i-1} \end{aligned} \tag{27}$$

This is substituted back into the weak formulation, to obtain a set of discretized equations. The test function, $v(x)$ is taken as $x^j,\ j = 1, 2 \ldots, p$.

$$\int_0^N (\sum_{i=1}^{p} i T_i x^{i-1} - \lambda \sum_{i=0}^{p} T_i x^i) x^j = 0, \text{ j = 1, 2... p} \tag{28}$$

Equation 28 is a polynomial equation and can be easily integrated to obtain a set of p equations. The resulting set of equations can be represented in matrix form as:

$$\mathbf{AT}_h = \mathbf{b}, \tag{29}$$

where A is called the *stiffness matrix* which contains the coefficients of T_i. b is a vector that is a function of T_i, also called the *load matrix*. The coefficients T_i are then computed using algebraic or numerical techniques for steady state problems, and numerical integration techniques for transient problems.

Almost all Computational Fluid Dynamics (CFD) tools use the finite element technique to solve thermal problems.

The disadvantage of the finite difference and finite element methods is that these methods are inherently slow. The speed and accuracy of these methods are dependent on the granularity of the grid/mesh that is used. A small increase in the size of the mesh makes the problem much more complicated.

2.8 Green's Functions

In the context of heat transfer, a Green's function is defined as the impulse response of a unit power source. The temperature profile for any power profile can be obtained by a convolution of the Green's function with the power profile. This can be represented by the following equation:

$$T = G \star P, \tag{30}$$

where $\star$ is the convolution operator. The convolution operation is as follows:

$$T(x,y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} P(x',y')G(x-x',y-y')dx'dy' \tag{31}$$

This follows by noting that any power profile can be represented as the sum of a large number of delta functions (unit power sources). The Green's function is the response to any one of these power sources (by definition). Hence the final temperature profile can be obtained by superimposing the numerous impulse responses. This task is achieved by convolution.

Green's functions have been frequently used to estimate the temperature profile in 2D chips. The primary advantage of this approach is speed. Unlike finite difference and finite element methods, that rely on the volume meshing of the entire multi-layer chip, Green's functions based approaches require meshing of only the power dissipating layers. This reduces the time required by this method by several orders of magnitude because modern chips have a complicated stacked structure.

To compute the convolution of two functions, the Fourier transform is often used. The 2-dimensional Fourier transform is given by Equation 32.

$$\mathcal{F}(g(s,t)) = \frac{1}{2\pi}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} g(x,y)e^{-j(xs+yt)}dxdy \tag{32}$$

The convolution of two functions in the time domain is equivalent to multiplication in the transform domain.

$$\mathcal{F}(f_1(t) \star f_2(t)) = 2\pi\mathcal{F}(f_1(t))\mathcal{F}(f_2(t)) \tag{33}$$

3 TAXONOMY OF SOLUTIONS

3.1 Scope of the Survey

In this paper, we cover the techniques for thermal estimation at the chip level developed in the last 15 years. We do not focus on the applications of these techniques to develop thermal management policies. Another category of techniques that we do not cover is thermal measurement based methods, the goal of which is to *measure* the temperature profile of a chip as accurately as

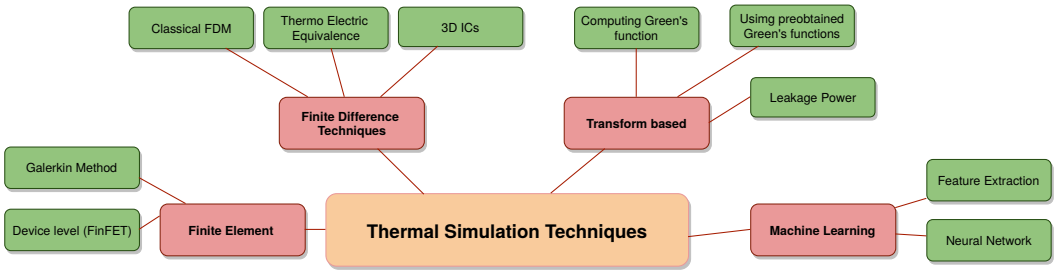

Fig. 4. Taxonomy of solutions

possible. There are two ways to achieve this. The first method is using thermal sensors placed at various locations on a chip. The second method is based on taking an infrared photograph of the die using a high resolution IR camera. These techniques have their own set of challenges and limitations. Because of space constraints, we focus on thermal simulation techniques that get their data inputs from the design of the chip rather than actual measurements (such as floorplan and power consumption). Hence measurement based techniques are out of the scope of our survey.

Although several techniques that we discuss are not inherently architecture level techniques, they have been widely adapted to compute the architectural thermal profile, and we refer to these as architectural thermal estimation techniques. An example of this would be the finite difference method. Almost all open source simulators use the finite difference method.

Another very relevant discussion would be about emerging technologies like FinFETs. As discussed in Section 2.5, the classic Fourier equation fails to hold at the device level, and the computationally expensive Boltzmann transport equation has to be solved. However solving the BTE for the entire chip is not practically possible. Hence researchers have proposed hybrid approaches that involve solving the BTE at dimensions smaller than the phonon mean free path, and the Fourier equation for the rest of the chip. These techniques are discussed in Section 4.

3.2 Classification Criteria

We classify the techniques used for thermal simulation into five broad categories: finite element based, finite difference based, spectral or transform based, machine learning based and thermal imaging based methods. A pictorial description of the space of techniques is given in Figure 4.

The finite element method is the most accurate albeit the slowest method for thermal simulation. It is mostly used by commercial simulators such as Ansys, COMSOL and FloTherm to obtain approximate solutions for heat transfer problems with complex shapes and boundary conditions. This method can incorporate material anisotropy and non-homogeneity. The accuracy of the solution depends on the discretization level of the domain and the number of elements. Creating a model for complex geometries requires some expertise, and is generally a tedious and time consuming process. The most important drawback is that the finite element method doesn't scale well, since a small increase in the size of the problem increases the simulation time several fold.

The finite difference methods are simpler to implement and quite accurate for a domain with regular shapes. However the accuracy for complex geometries is lower than that obtained using finite element methods. The most widely used algorithm in finite difference methods is to create an analogous electrical RC circuit where resistance and capacitance are calculated from the material properties. Several widely used simulators have been developed based on this method. These

simulators have been augmented to be able to solve the heat equation when microchannel cooling is deployed. They have been further accelerated by implementing on a GPU.

Spectral or transform based techniques are typically based on the Green's functions. The Green's functions are computed by applying a unit power source (a Dirac delta function) to the center of the chip. The resulting temperature profile is measured and stored as the Green's function. This part is offline. In the online part, the power profile is convolved with the Green's functions to obtain the temperature profile. Since the convolution operation can be achieved by a Fast Fourier Transform (FFT) and inverse FFT operation, the complexity of the problem is much lower than finite difference based methods. It is possible to further optimize this process. The primary advantage of the transform based techniques comes from the fact that the entire package is not meshed, since the exact heat transfer path is not modeled. Only the layers containing the heat sources are meshed, so the complexity of the problem is much lower than finite difference or finite element based methods.

In machine learning based techniques, a model is trained and then used to predict the temperature profile. First, the temperature profiles corresponding to a set of power profiles are obtained. These are then used to create a model using machine learning techniques. The model is trained using these values. The training phase can be very long, but since this part is offline and needs to be done only once, it is not a limitation. Subsequently the temperature profile for any power profile can be quickly calculated using the model.

Each category of techiques has two important characteristics: steady state or transient thermal model and the ability to model leakage power. With each technique that we discuss, we also specify these two characteristics. A **steady state analysis** gives the final thermal profile for a given power profile. However, when the the power itself is a function of time, or when we are interested in finding the temporal evolution of temperature, a **transient analysis** is necessary.

4 FINITE ELEMENT BASED TECHNIQUES

The basic principles of finite element based techniques have been discussed in Section 2. When the finite element method is applied to thermal estimation, the basic steps followed are:

(1) The chip is divided into a mesh, and the element size is carefully selected to keep the problem complexity manageable.
(2) The physical properties (thermal conductivity and specific heat capacity), floorplan of the chip and the boundary conditions are taken into account and a set of partial differential equations are formulated.
(3) The equations are solved using the Galerkin method. A numerical technique such as the Runge Kutta method is employed to find the solution.

Zjajo et al. [102] use the finite element method to estimate the thermal profile of a 3D chip. They apply the Galerkin method to solve the heat equation after applying the surface boundary condition residual, and arrive at Equation 34.

$$\begin{aligned} &\int_V v C_V \frac{\partial T_t}{\partial t} dV + \int_V (\nabla v)\kappa(\nabla T) dV - \int_V v Q dV + \int_S v h (T - T_a) dS = 0 \\ &v(x) = 0,\ x \in S, \end{aligned} \tag{34}$$

where v is the test function used in the Galerkin method, V is the volume of the domain, T_a is the ambient temperature, and S is the surface of the domain. The authors then apply the boundary conditions and make some simplifying assumptions. They then discretize the equations and use time step marching techniques to obtain the solution of the resulting equations. The authors apply a modified third order Runge Kutta scheme. To accelerate convergence, the solution is obtained for

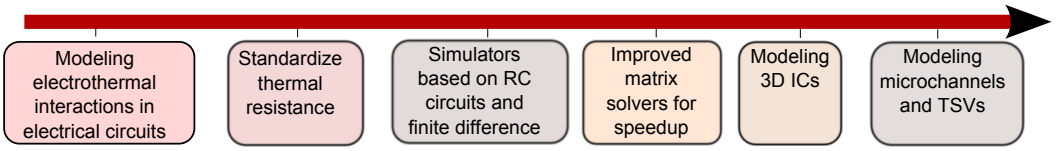

Fig. 5. Evolution of finite difference based techniques

a coarse mesh initially and then refined multiple times. The forward and backward Euler methods are used to resolve circular dependency problems, and improve efficiency.

To evaluate this method, Zjajo et al. consider a double layered die stack with a fixed heat sink. The authors demonstrate that their method is 1-2 orders of magnitude faster than other state of the art methods. Also, their modified Runge Kutta solver outperformed solvers based on extant Euler and Newmark methods.

4.1 Device Level Thermal Modeling Using FEM

The finite element method has also been used to calculate the full chip thermal profile using a hybrid Fourier-Boltzmann method. Allec et al. [3] propose an integrated solver to estimate the full chip profile as well as the device level profile accurately. At distances greater than the mean free path of phonons, the Fourier equation is used to compute the full chip temperature profile. At the device level, they solve the steady state Gray-Boltzmann equation under the relaxation time approximation. The Fourier and the BTE solvers are invoked iteratively, and the heat flow and boundary conditions are updated after each iteration. The tri-diagonal matrix algorithm (TDMA) is used to solve the Fourier and the Boltzmann equations. They employ a hierarchical spatial partitioning algorithm to reuse the fine grained partitioning method and save memory. To validate their Boltzmann solver, the authors solve the Heaslet and Warming problem [29] and report good agreeement with standard results. Next the authors show that using the Fourier model at the device level introduces errors over 30% compared to the Boltzmann method, and the error increases as the device size decreases. They also show that when the inter-device distance is lesss than 20 nm, the Fourier method is quite accurate with an error of less than 1%.

Subsequently the authors use the Boltzmann method over the entire chip. This takes 16.3 hours to solve. They then use their hybrid technique and report speedups in the range of 23× to 150×, with an error of 4%. However, they do not model the substrate beyond the mean free path of phonons. The simulation time for modeling a 150 million transistor device is reported to be 167 minutes for bulk silicon and 179 minutes for FinFETs. The authors model leakage power iteratively.

Hassan et al. [28] extend this work and incorporate a lookup table to store device level temperature information. They further refine the spatial granularity for the Fourier solver if the temperature difference between neighboring elements is greater than a threshold value. The authors validated their results against COMSOL, and found the average error for 17 benchmarks from the SPEC suite to be 1.77%.

5 FINITE DIFFERENCE BASED TECHNIQUES

5.1 Simulators Solving the Heat Conduction Equation

In these techniques, the solution to the heat conduction equation is found analytically using the finite difference method. Figure 5 descibes a timeline of the evolution of finite difference based approaches.

In 2001, 3D thermal ADI was proposed [89], which solved for the temperature distribution of the complete chip using the heat conduction equation. The conduction equation was first discretized in space and then in time. To obtain the transient temperature profile, the time step from n to $n + 1$ was divided into three steps, and the alternating direction implicit method was used along each direction. The authors analytically solved the resulting equations for the non-homogeneous case as well, where the chip layer consists of multiple materials including a single active layer of silicon. The convection boundary conditions due to the package and heat sink were modeled as resistances on the six sides of the chip.

To incorporate leakage power, they re-iterate multiple times until the temperature and leakage power values converge.

Yang et al. propose ISAC [92], in which they dynamically adjust the spatial and temporal resolution to achieve high speeds without a loss in accuracy. They also model the dependence of thermal conductivity on temperature. The authors use multigrid methods with iterative relaxation methods to solve the heat equation. First a coarse simulation is performed. Then the authors refine the resolution iteratively for nodes for which the temperature difference between adjacent nodes exceeds a threshold. For the transient simulation, the authors use asynchronous time marching techniques. In these techniques the time step size for each thermal element does not have to be the same. This results in improved transient simulation efficiency. To validate their results, the authors use two chip stacks and model their thermal behavior using ANSYS. Compared to conventional multigrid approaches, they obtain a speedup of upto 690 times for the steady state case. For the transient case, they achieve two orders of magnitude speedup over fourth order Runge Kutta based solver.

5.2 Simulators Invoking the Analogy to Electrical Techniques

The standard approach here is to construct an electrical RC circuit based on the parameters of the chip. The chip properties are used to calculate the equivalent resistance and capacitance, and the resulting RC circuit is simulated using standard circuit simulation methods.

This method has evolved over several years (see Figure 5). The early efforts were directed at modeling the electro-thermal interactions in electrical circuits, similar to the SPICE simulator [24]. Since SPICE had efficient routines for solving the electrical parameters in RC circuits, these methods used similar approaches to solve for the thermal properties as well.

5.2.1 Early Efforts. Fukahori and Gray [24] proposed a thermal simulator that considered the effects of thermal gradients on performance in a chip package structure. They developed a physical model of the chip package structure. The model was then converted into an electrical circuit by discretizing the volume into subregions connected by a thermal resistance on all sides, using a finite difference approach. The thermal capacitance of the remaining structure was lumped and added to the outside edge. Then they formulated the matrices corresponding to the electro-thermal interaction, and solved them using the Newton-Raphson method. To calculate the transient response, they used trapezoidal integration. This was a very early and simplistic approach, whose primary contribution was to provide a modeling framework for electro-thermal interactions.

Lee and Allstot [44] in 1993 proposed a thermal simulator that was then coupled with the SPICE electrical simulator to model the electro-thermal interactions. They obtained the power dissipation from SPICE, and used it in the heat conduction equation to determine the temperature gradients.

By 1995, it was common for manufacturers of electronic components to provide the thermal properties of their devices in the form of a thermal resistance [43]. In efforts to standardize this method, authors [43] used a three resistor model for the chip, connecting a junction node to the top, bottom and side nodes of a chip. To compute these resistance values, they used a full chip model

in a circuit analysis tool and imposed boundary conditions on the faces to calculate the junction temperature.

In 1998, Cheng et al. [13] proposed a chip level electro thermal simulator, ILLIADS-T that modeled the steady state thermal profile, along with its performance. It took as inputs the layout of the chip, the parameters defining the packaging, and periodic input power values. To calculate the power dissipated by each gate, it uses a timing simulator, ILLIADS [71]. ILLIADS-T iteratively fed the power values obtained to the thermal simulator, to obtain the updated temperature values. These values were used to update the model parameters, which in turn were used to update the temperature values. To obtain the thermal profile, the 3D heat diffusion equation was solved for the substrate, while the package and heat sink was modeled as a 1D thermal resistance. The resulting equations were solved using a numerical 3-D finite difference approach. Li et al. [48] propose an iterative multigrid method to model a full chip comprising of the substrate, polysilicon, metal and ILD layers. Since the thermal conductivity of a layer may not be homogeneous they divide each layer into bins, and calculate the equivalent thermal conductivity of each bin. They define appropriate coarser grid operators, interpolation and restriction operators. The authors do not apply coarsening in the z-direction except in the silicon substrate, and keep at least two grid points in the z direction in each layer. An approximate solution is obtained for the coarse grid, which is then mapped to the finer grid using an interpolation operator. This interpolation operator is arrived at considering the discontinuities at material boundaries. Finally they use a vertical line smoother to improve convergence.

5.2.2 The HotSpot Thermal Modeling Tool. The most popular thermal simulator in this area is the HotSpot temperature modeling tool [30, 73, 98]. This was the first work to propose a detailed thermal model, which accounted for all the layers in a 2D chip. It provides temperature values at the granularity of micro-architectural units. It is parameterized, which makes it well suited for architectural exploration. The chip here is assumed to be a stacked layer structure. HotSpot takes the floorplan of the chip as input, which contains a description of the microarchitectural units, and their locations. It also takes the power dissipation of each of these units over a time step as input. Based on these values, it generates a thermal RC network. The RC model has four layers. Three of these are conductive layers corresponding to the die, the heat spreader and the heat sink. The fourth layer is convective and corresponds to the heat sink to air interface.

HotSpot discretizes each layer into a number of blocks and calculates the lateral resistance between two adjacent blocks, and the vertical resistance between blocks in different layers. The resistance, R is given by:

$$R = \frac{t}{kA}, \tag{35}$$

where A is the area, t is the thickness of the layer, and k is the thermal conductivity per unit volume. The blocks in the die layer correspond to microarchitectural units (in whole or in part). Modern chips have a pyramid structure, with the heat spreader being larger than the die, and heat sink in turn being larger than the spreader. HotSpot discretizes the spreader into five blocks – one corresponding to the area directly under the die, and the remaining four are trapezoidal blocks connecting the corners of the die to the corners of the heat sink [30]. The heat sink is discretized similarly. The convective layer is modeled by a single thermal resistance. There is a thermal resistance between the heatsink and the air due to convection, which is computed using Equation 16. The heat transfer coefficient h is obtained from the heat sink specifications.

The temperature of the ambient air is considered to be a constant. The power dissipated by the microarchitectural units are modeled as a current source in the center of the unit.

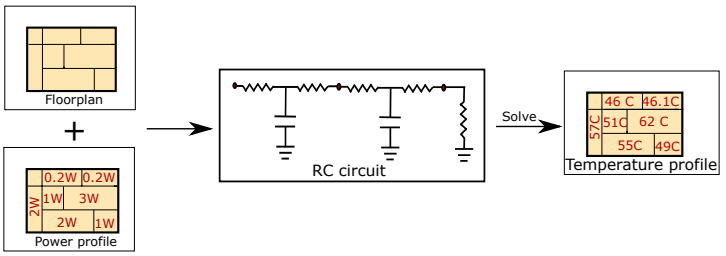

Fig. 6. Modeling technique of HotSpot

The thermal resistance is obtained by computing the reciprocal of thermal conductance as given by Equation 14. The convection resistance is taken to be a constant value of 0.8 K/W. HotSpot (see Figure 6) also has a vertical thermal capacitance between each node and the ground. This is calculated using Equation 18. The thermal properties of silicon are used to compute the values of the various constants. Thus HotSpot is capable of modeling the transient response in addition to the steady state response. Differential equations are then solved using a Runge Kutta method of order 4. More complicated models consider heat transfer via the package as well.

The results were initially validated with a commercial FEM tool FloWorks ([73]). The power values were obtained from Wattch ([9]), an architectural level power analysis tool. Every time the solver was invoked, it took 50 μs, on a 1.6 GHz AMD Athlon processor. This means an additional simulation time of less than 1%. The ambient temperature was taken to be 45°C . The maximum error obtained using HotSpot as compared to the FloWorks simulation was limited to 5.8%. Subsequently, results were validated with the Xilinx Virtex-2 Pro FPGA by using ring oscillators as temperature sensors. For six sensors placed across six functional units on the FPGA, the HotSpot tool's results have an average error of less than 0.2°C (10%).

The HotSpot thermal simulator has been widely used to estimate the temperature profile for several architectural techniques. Hung et al. [35] use the HotSpot tool to carry out temperature aware floorplanning using genetic algorithms. [32] use HotSpot to propose a temperature aware algorithm for IP virtualization and placement for a generic network on chip [32].

Han and Koren [25] also use HotSpot to evaluate their temperature aware method for floorplanning based on an algorithm called *simulated annealing*. However, the simulated annealing algorithm requires quickly sifting through millions of potential floorplans. This makes it prohibitively expensive to compute the temperature profile for each such floorplan. Hence they devise a metric to serve as an estimate of the maximum temperature in the chip. This metric is based on the heat diffusion between two adjacent blocks [25]. They estimate this heat diffusion to be proportional to the difference between power densities of adjacent blocks multiplied by their shared length. The heat diffusion for an architectural block is obtained by summing the inter-block heat diffusion values over all its neighbors.

Han et al. [26] make simplifications to the state equation for discretized power inputs by noting that the power vector remains constant within a sampling interval. They use HotSpot and apply a single input and calculate its step response, which is then used to calculate the state matrices. Then they directly calculate the temperature for a small time interval without using the Runge Kutta solver. For further speed up, they split the algorithm into offline and online steps. The offline part is precomputed and saved in a table. In the online part the input power trace is divided into windows and the temperature profile is calculated using a convolution operation. They call this method CONTILTS or convolutional TILTS. TILTS was found to be upto 1300 times faster than HotSpot. CONTILTS achieved further speed up (6000 times), while maintaining the same accuracy. However

their method is only for discrete power inputs. It also does not scale well for higher number of nodes and will require employing model order reduction techniques.

5.2.3 Miscellaneous Techniques. SESCTherm [58] is another thermal simulator having three components: the finite difference modeling framework, the thermal modeling framework, and the material modeling framework. The material modeling framework updates the material properties based on the changes in temperature. SESCTherm partitions the chip and package into several nodes. They have a material model, which models the interconnect, package and substrate thermal properties. They then create lumped thermal characteristics and distribute them across the chips. The material properties are updated based on the temperature. The original purpose of this paper was to study the effect of various components of a package on the thermal profile.

Li et al. [46] propose a mathematical thermal simulation algorithm, ThermPOF that uses measured or simulated temperature and power data for training, and determines the temperature for a variety of power configurations during testing. They first find out the impulse response matrix (which is a function of time) from the step response, by differentiating it. The authors then sample the impulse response using the logarithmic scale, since temperature rises rapidly initially and slows down later. They perform corrections on the sampling start and end times to ensure stability. This is then transformed to the Laplace domain to find the poles and zeros of the system using the generalized pencil of function method. A recursive convolution is used to compute the temperature profile in $O(nq)$ time, where n is the number of time segments and q is the number of poles of the transfer function. To futher reduce the order, the Krylov subspace based method PRIMA is used. The authors achieve steady state errors of less than 0.4% and transient errors of less than 6%, compared to empirical data obtained from Intel. The runtime of their algorithm is 1.3s when the reduced order is set to 50, and 0.8s when the order is 30.

In [47], the authors enable parameterized modeling, by building *response surface models (RSM)* for each time point and each parameter variable. These RSMs map several input variables to one or more output or response variables. Next, the authors use the ThermPOF algorithm over each coefficient of the RSM, to obtain the transient thermal profile.

5.3 Simulators for 3D ICs

Hung et al. [34] extended the HotSpot thermal modeling to 3D chips in their tool called HS3D. Their model consists of several layers of silicon in between which there is an interlayer glue material. For obtaining the solution, they used the HotSpot tool. Since then, this method has been incorporated in HotSpot itself [31].

3D chips have a severe temperature issue and their performance is limited by this factor. To alleviate this problem, researchers have suggested etching microchannels in the chip to carry a cooling fluid. Since this seemed to be the most practical solution to the cooling problem in 3D chips that did not sacrifice performance, it has caught the attention of several researchers.

5.3.1 Thermal Wake Functions. Mizunuma et al. [57] identify a phenomenon called *thermal wake functions* that complicate the temperature profile calculation in the presence of fluid carrying microchannels. The temperature of the fluid upstream near the heated die is significantly high. As the fluid flows downstream, heat dissipates quickly and the temperature decreases exponentially. Similarly, the fluid in the channel heats the side walls, which in turn heats the fluid in other channels. So they consider two types of wake functions: a downstream function and a transverse-to-flow thermal wake function. These wake functions are considered to be equivalent to thermal resistances in an RC network. The different components of the temperature rise are expressed using different wake functions. These are then calculated empirically by setting boundary conditions for the grid cells appropriately. Once the resistances corresponding to the wake functions are obtained, they

are added to the conventional thermal resistance in an RC circuit and solved using matrix solvers. They validate their results against a commercial tool called *Zflow*. They achieved a 400X speedup over Zflow with an error of 2%.

Shi et al. [70] capture the thermal wake effect in their microchannels thermal model by using three resistances: conduction, convection and fluid flow based. They use this thermal model to propose a non uniform microchannel placement algorithm to save pumping and cooling costs. They iteratively remove microchannels with the lowest cost and estimate the new temperature, until this temperature is less than a threshold.

Kim et al. [39] study the effects of single phase versus phase change cooling, and conclude that two phase cooling performs better. They use the energy and momentum conservation equations to model thermal and fluid flow.

5.3.2 Modeling Microchannels and TSVs into HotSpot. Coskun et al. [15] extended the HS3D modeling technique to incorporate the effects of through silicon vias (TSV) and microchannels. In the multilayer model of HotSpot, they incorporate an additional layer to model TSVs and microchannels. Instead of a uniform thermal resistivity for the interface layer, their model has a varying thermal resistivity for each grid cell, which can be set at runtime. To model TSVs, they adapt the conductivity of the grid cells at the TSV locations, based on the properties of the interface material and TSVs. For grid cells at the microchannel locations, they modify the resistivity on the basis of the fluid flow rate. The authors conduct three experiments to determine the optimal granularity of modeling needed. In the first experiment, the authors assume a uniform TSV distribution across the die. In the second experiment, the authors assume a specific TSV distribution per block (i.e., core, cache, crossbar). In the third experiment, the authors use the exact locations of the TSVs. They conclude that the accuracy obtained by the block level modeling of TSVs is similar to using the exact locations, with a much lower computational overhead. They used this modeling technique to compare a system with microchannel cooling to heat sink based cooling where they use HotSpot with default parameters as the thermal modeling tool.

5.3.3 3D ICE. Sridhar et al. [76, 78] proposed to incorporate the effect of microchannels on the thermal profile by adding a convective term for the heat exchange by fluids. Their tool is called 3D-ICE. Since the heat transfer by a flowing fluid is not isotropic, the additional term is modeled as a temperature controlled heat source. The conduction term is neglected for the microchannel cell, and the convection term translates to a voltage controlled current source in the RC circuit. The authors calculate the interface temperature at the boundaries of the microchannel cells by interpolating the node temperatures of the two adjacent cells. The next task is to compute the four conductances for the microchannel cells, which result in heat transfer from the walls to the fluid. These are calculated using an empirical formula. The capacitances are calculated just like solids. These microchannel cells are connected to silicon cells similar to conventional finite difference modeling techniques. The backward Euler method is used for integration. The KLU and SuperLU libraries have been used as the solver.

To verify their results, the authors used the ANSYS CFX tool. The authors have implemented two test stacks - the first one had two active silicon layers with one microchannel layer in between, while the other one had three active layers with four microchannel layers. Both the dies are $1\ cm \times 1\ cm$. In the first case, a uniform heat flux was applied to both the dies for $0.1\ s$. The temperature difference between that obtained using 3D ICE and Ansys CFX was limited to 1.6°C, or 3% of the peak temperature rise. In another experiment, the hot spot switching activity was simulated for $0.6\ s$. The experiment was repeated with different flow rates. The error was limited to 1.5°C or 3.4% of the peak temperature rise. Their model was also found to be 200-1000X faster than CFX simulations.

Qian et al. [64] improved this method by noting that the heat removal because of TSVs is anisotropic in nature. The conductivity improvement is more in the vertical direction than laterally. They also note that there is a higher rate of heat transfer at the microchannel inlet. They modeled the effect of TSVs, heatsinks as well as microchannels. They assume the TSVs to be rectangular in cross section, and modified the conductivity of each grid cell based on the location and properties of TSVs. The TSVs have a dielectric liner with low thermal conductivity. This is accounted for in the grid cell conductivity in the presence of TSVs. First the conductivity of grid cells is calculated without TSVs. Then the effect of TSVs is incorporated. If multiple TSVs are present in a grid cell, their effects are superimposed. After the conductance matrix G is constructed, the power values are read to construct the power matrix P. Then the temperature profile is calculated using the equation, $GT = P$, by the KLU solver. The complexity of the model is $O(N)$, where N is the number of grid cells. The results are validated against COMSOL for a 2 layer chip stack. A simple 3×3 floorplan and a 16×16 grid size is used. For microchannels based cooling, the authors report a maximum error of 1.1%, and for heat sink based cooling, they report a maximum error of 1.2%. The error obtained using 3D ICE for their model was higher (up to 20% in some cases). A speedup of 21X was obtained compared to HotSpot for the heat sink based model.

5.3.4 GPU Based Approaches. Feng and Li [20] proposed a GPU based thermal simulator for microfluidic cooling. They start with existing microchannel modeling techniques, and try to adapt it to a GPU. They adopt a two step block based iterative method. In the first step, the solution is updated in the vertical direction. In the second step, the solution is updated in the horizontal direction along the direction of fluid flow. They use data structures that exploit the parallel computation capabilities of a GPU. They implement four preconditioned conjugate gradient iterative solvers for the GPU. These solvers are compared with CPU based solvers and direct matrix solvers. The direct solvers were not able to process the complex thermal circuits except for the ones with small grid sizes. The GPU based iterative solvers could solve a thermal problem with 6 million nodes in 10 seconds. A 35X speedup was obtained by the proposed solvers over CPU-based solvers that iteratively find the solution, and a 360X speedup was obtained over the CPU-based direct matrix solvers. However this model can only calculate the steady state temperature profile.

Liu et al. [51, 52] use the 3D ICE thermal modeling technique and solve it using a GPU accelerated GMRES solver. It employs the Krylov subspace based method with pre-conditioning. The authors identify computation intensive methods in the GMRES method and parallelize it for a GPU. For pre-conditioning they use the asymmetric Bridson Approximate Inverse pre-conditioner. They also implemented GMRES on a CPU with Jacobi iteration. They observed that GPU-GMRES is faster than the CPU GMRES that had been parallelized, by 4.3X for the steady state problem. For the transient case, it was 1-2 orders of magnitude faster.

5.3.5 Miscellaneous Techniques. Fourmigue et al. [23] use the implicit method to discretize the finite difference equations obtained for a microchannel based chip that may have TSVs. They use an adaptive grid size such that a cell is assigned to the largest homogeneous block of a material. The discretized equation is split into three stages using the splitting operator technique. The tridiagonal matrices obtained are solved using the Thomas method [12]. For further speedup, independent tridiagonal matrices are solved in parallel. They validate their approach against 3D ICE. They report a two orders of magnitude speedup with an error around 1°C.

In [21], the authors try to reduce the splitting error. They use the explicit forward Euler method and obtain an error within 1% as compared to 3D ICE.

However, with a detailed TSV model, the number of iterations and the simulation time becomes very large. Formigue et al. then propose another technique [22] based on the D'Yakonov scheme [19]. The fundamental heat transfer equation is discretized in time using the Crank Nicolson [17] scheme.

Then in accordance with the D'Yakonov scheme, a perturbation term is added and the equations are simplified. The resulting equations are solved using the Thomas algorithm [12]. They compare their proposed method with their earlier work [23] and 3D ICE. The new method was found to be 4-5 times faster than the earlier method and orders of magnitude faster than 3D ICE.

5.4 Modeling Leakage Power

Wang et al. [87] propose an analytical leakage aware thermal estimation technique for 2D chips. They consider subthreshold leakage as a linear function of temperature around a local reference temperature. The reference temperature is updated when the node temperature differs the reference temperature by a fixed value (taken as 10°C). The authors use the sampling based model order reduction technique to reduce computations. They choose low frequency sampling points and then perform singular value decomposition on the resulting matrix. They then retain the first q columns and discard the remaining values to obtain the projection matrix, since the columns are ordered by importance. To account for the changes in projection matrices in transient thermal estimation, they add a correction factor. To further reduce computations because of changes in the reference temperature, the authors update the projection matrix only when the estimation error exceeds a threshold. When the projection matrix has to be updated, the authors propose a method to partially update it. This helps in reducing the number of times the whole model order reduction process needs to be repeated. To evaluate their results, they use four chips having 9-36 cores. The authors compare their results against iteration based method and TILTS [26]. They concluded that their local linear leakage thermal models were quite accurate with errors within 0.12°C . The model order reduction technique introduced a very small amount of error (0.3°C). In terms of speed, the linear leakage model was faster than TILTS, which were both much faster than the iteration based method. Using model order reduction, further speedup was obtained.

Yan et al. [90] propose a leakage aware analytical thermal modeling technique for 3D chips. They also assume a linear leakage model. The authors demonstrate their method for the steady state. Their leakage model uses a different expansion point for each grid in the model. A coarse thermal simulation is first performed to estimate the expansion points. To further improve the accuracy, they introduce an additional term for higher order coefficients of leakage power. This term is determined iteratively. The authors show that this method is equivalent to the Newton Chord method [59] having a linear convergence rate [42]. To solve the resulting equations, the author use the aggregate-based algebraic multigrid preconditioner (AMG). This is used in conjunction with Conjugate Gradient method [59]. The authors show that using a uniform linear leakage model, an error of upto 12.67K can be obtained. The authors also show that even the linear model with multiple expansion points has large errors, while their corrected linearized model is very accurate. Their AMG based solver achieves a speedup of 12 times over HotSpot in microchannel-cooled 3D ICs and 139 times in heatsink-cooled 3D IC.

6 SPECTRAL OR TRANSFORM BASED TECHNIQUES

The primary advantage of spectral or transform based techniques is that they are significantly faster than finite difference and finite element based techniques, at the cost of a small amount of accuracy. Figure 7 shows a generic transform based technique.

These approaches are based on the Green's functions. Once, the Green's functions have been obtained, a simple convolution with the power profile will generate the temperature profile. The difficulties are in computing the Green's function for a given geometry, and in incorporating boundary conditions correctly.

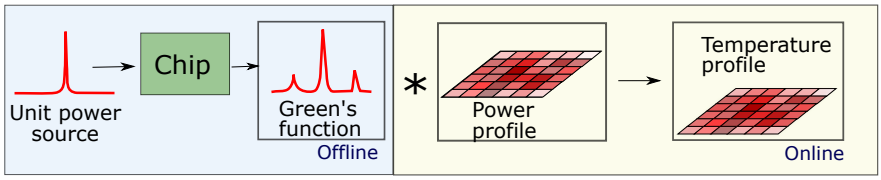

Fig. 7. Transform based technique

6.1 Analytically Computing the Green's Functions

These category of techniques start with the heat conduction equation, and analytically compute the Green's functions. However, the drawback is that they are not able to compute the transient temperature profile because of complexity issues.

Wang and Mazumder [86] computed the Green's functions analytically in their technique called LOTAGre. The model had multiple layers as in a standard 2D chip and incorporated the boundary conditions for the side walls, including the convection effects from the heat sink to the ambient. They also studied the effects on non-uniform ambient temperature surrounding the chip. The temperature distribution is calculated in two parts: inhomogeneous temperature profile because of sources within the chip and a homogeneous temperature profile due to the temperature around the chip. The inhomogeneous part is obtained by the convolution of the Green's function with the power profile.

The final temperature profile is obtained by superimposing the homogeneous and the inhomogeneous components.

Zhan and Sapatnekar [94] used the Discrete Cosine Transform (DCT) to transform the power profile to the frequency domain and compute the Green's functions analytically. The heat conduction equation was simplified by assuming a steady state homogeneous condition with no power sources in the chip, to obtain:

$$\nabla^2 T = 0 \tag{36}$$

Then they imposed adiabatic boundary conditions on the side walls, a convection boundary condition on the bottom surface (near the heat sink) and a conductive term at the top of the chip. This resulted in a system of equations involving the Green's functions and the boundary conditions. Using the method of separation of variables, they arrived at a double infinite summation. With further modifications, the final temperature profile was reduced to 1D and 2D DCT sequences. They stored these in a lookup table to further speed up computations. For computing the temperature profile from a given power profile, they looked up the values of the DCT sequences, and obtained the final temperature profile by summing the results of convolutions between the power profiles and the Green's functions.

The authors [95] further improved this method by performing the final temperature evaluation step in the frequency domain itself. In the first step, the power map was transformed to the frequency domain using a 2D DCT. Then the Green's functions were computed using the method outlined earlier, with some modifications to remain in the frequency domain. By multiplying the power profile and the Green's function in the frequency domain, the frequency domain representation of the temperature profile was obtained. Finally an inverse DCT was used to obtain the required temperature profile. However, they used a single layer thermal model, consisting of one homogeneous material.

Oh et al. [60] calculated the Green's function for a single component of the chip (such as interconnects) by applying a similar technique. They separated the Green's function by the separation

of variables method in each direction, and used a Fourier expansion in the x and y directions. The computational complexity of this algorithm is $O(n * l)$, where n is the number of nodes and l is the number of layers.

Qian and Sapatnekar [65] note that a major problem with the technique proposed by [96] is that a practical pyramidal chip-spreader-sink structure cannot be modeled. Hence they propose an eigen decomposition based fast Poisson solvers for modeling the pyramidal structure. They consider the approximate solution given by the fast Poisson solver (FPS) with rectangular domain as the preconditioner for Preconditioned Conjugate Gradient (PCG). It then iteratively converges to a more accurate solution. They also show that the FPS is equivalent to the Green's function solution discretized in the frequency domain, when evenly spaced sampling is done in the spatial domain.

6.2 Using Pre-Obtained Green's Functions

While calculating the Green's functions analytically, some complex issues arise because of the geometry of the chip, which the analytical approaches are not able to incorporate [101]. Hence later techniques rely on FEM software to compute the Green's functions [56, 101]. Since the Green's functions need to be computed only once and can be stored, the higher computational time of the FEM technique is not a problem, especially since it leads to improved accuracy. The storage overhead is also not very large, since typically a 16×16 or a 32×32 grid per active layer is sufficient to accurately estimate the temperature profile.

The authors of the Powerblurring method [101] used the ANSYS simulator to obtain the Green's function. They treated the Green's functions as a thermal mask, which has to be applied to the power profile to obtain the temperature profile. They applied two corrections to this mask: the first was to modify the method of images to remove the corner and edge effects in a finite sized die. The authors replaced the adiabatic boundaries of the chip with an image source. The final temperature profile is calculated using the central portion (two thirds) of the resulting enlarged matrix. The second correction was that the authors claimed that the pyramid shape of the package results in the Green's functions being incorrectly calculated for edges and corners. To account for this error, they obtained the temperature profile for a uniform power distribution, calculated the error and scaled the Green's functions accordingly to compensate for this error.

To compute the transient temperature profile corresponding to a time dependent power profile, they applied a delta function to the center of the chip for a short duration and measured the temperature profile at multiple time steps that were smaller than the delta function width. A series of thermal masks were obtained corresponding to each point in time. In an earlier version of the Powerblurring technique [61], the final temperature profile was calculated by discretizing the power profile into a series of delta functions corresponding to each time step. Then, the thermal mask obtained from the delta power source was convolved with the power values at each time step. If there are n time steps, n such convolution operations are done. These values were stored as the transient basis. To compute the thermal profile corresponding to a long power pulse, the transient basis was shifted one step at a time, and normalized with the power values. Finally they superimpose all the transient bases obtained in such a manner. As can be seen, this method is computationally expensive.

In a more recent paper [101], the method is improved by superimposing the effect of all the previous thermal masks that have an effect in the current time interval. Thus the sum of several convolution operations is still needed, but there is no need to store the transient basis.

Touzelbaev et al. [81] compute the device level thermal profile using a similar method. However, instead of using convolution directly to compute the final temperature profile, they propose an Infinite Impulse Response (IIR) filter based approach. The transfer function of the filter is discretized and the inverse z-transform is used to obtain the transient temperature response.

6.3 Modeling Leakage Power

Sarangi et al. proposed the simulator Lightsim [68] in which they made two fundamental improvements. The first was to improve the speed of the convolution operation by using the Hankel transform instead of the Fourier transform. A 2D Fourier transform is equivalent to a zero order 1D Hankel transform for a radially symmetric function. In an infinite sized die, the temperature profile is radially symmetric. Using this approximation, they converted the 2D Fourier transform operation traditionally used to a 1D Hankel transform. They make some corrections for edges and corners though.

The second improvement that they provided was an analytical technique to incorporate the effects of leakage in the Green's functions. Based on this, they formulated a dependence relation between the final temperature value, the Green's function without considering the effects of leakage and the leakage power. The leakage was assumed to be a linear function of temperature as given by:

$$P_{leak} = \beta T, \tag{37}$$

where β is the temperature dependence of leakage power.

A unit power source was applied to the grid cell at the center of the die. The resulting temperature rise, T was found to be dependent on the heat spreading function, f_{sp}. Using Green's functions, the temperature rise is given by:

$$T = T_F - T_0 = f_{sp} * (P_{dyn} + \Delta P_{leak}) \tag{38}$$

$$T = f_{sp} + \beta f_{sp} * T \tag{39}$$

They assume that the heat spreading function f_{sp} has two components: the first component is a rapidly decaying function called f_{silic}. A small amount of heat is transferred to the heat spreader, which is captured by the constant κ. The authors used these in the above equations and made some simplifying assumptions. The equations were solved analytically with the help of the Hankel transform. The Hankel transform helps in converting the convolution operation to multiplication, resulting in an algebraic equation in the transform domain, while requiring only a 1D transformation. After a few simplification steps, the analytical steady state temperature profile in the presence of leakage is obtained. This gives the leakage modified steady state Green's functions. To calculate the transient temperature profile, they introduced a thermal capacitance, C:

$$T = f_{sp} + \beta f_{sp} * T - C f_{sp} * \frac{\partial T}{\partial t} \tag{40}$$

This procedure resulted in a differential equation which was solved using some more approximations, and the leakage modified transient Green's functions were obtained. To correct for the boundary effects, they made an assumption. The impact of the neighboring cells' contribution to the temperature profile at the edges is one fourth of the size at the center. Similarly, it is halved at the edges. Hence, they divided the leakage term by 4 at the corners (2% of the area) and 2 at the edges (8% of the area). The 2D FFT of these three functions was computed and multiplied with the 2D FFT of the power map. The final temperature profile is obtained by adding the three temperature profiles: center, corners, and edges.

The authors claimed that their simulator is 3500X faster than HotSpot, with an error less than 2% in the steady state case and 5% in the transient case.

Sultan et al. [80] extended this idea to 3D chips. In a 3D chip there is a complex interaction between the temperature profile of a layer and the power sources present in each layer. They

modeled this interaction using a system of equations:

$$\begin{aligned}\mathcal{T}_{1k} =& fsp_{1k} + \beta(fsp_{11} \star \mathcal{T}_{1k} + fsp_{12} \star \mathcal{T}_{2k} + fsp_{13} \star \mathcal{T}_{3k} + \\ & \ldots + fsp_{1l} \star \mathcal{T}_{lk}) \\ & \ldots \\ \mathcal{T}_{lk} =& fsp_{lk} + \beta(fsp_{41} \star \mathcal{T}_{1k} + fsp_{42} \star \mathcal{T}_{2k} + fsp_{43} \star \mathcal{T}_{3k} + \\ & \ldots + fsp_{4l} \star \mathcal{T}_{lk}),\end{aligned} \tag{41}$$

where $\mathcal{T}_{ik}$ represents the temperature rise in layer i because of a source in layer k.

They computed the Fourier transform of the set of equations, and solved them using Mathematica with the help of some approximations. To further speed up the computation, they used the Hankel transform instead of the Fourier transform, and stored the solution in the transform domain itself. This saves an additional step of calculating the Fourier transform of the Green's functions for the final temperature. For correction at the edges and corners, they added the values at the opposite corners.

To compute the transient temperature profile, a 3D version of Equation 40 was used resulting in the following set of equations:

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

To solve these equations, they again computed the Hankel transform and converted the differential equation into algebraic equations using the Laplace transform. Ignoring the secondary effects, and with further simplifications, they arrived at the leakage modified transient Green's functions. To further speed up the temperature profile calculation, they used the discrete Hankel transform, and pre-computed and stored the results in a lookup table. For the steady state case, they obtained an error of less than 0.46°C using the Fourier transform and 1.5°C using the Hankel transform with a speed up of 68X over 3D ICE. For the transient case, the error obtained was less than 6%, with a 71X speedup over 3D ICE.

7 MACHINE LEARNING TECHNIQUES

These techniques employ machine learning algorithms to train a model, which is then used to predict the temperature profile.

The steps employed here are outlined below:

(1) A large amount of training data is collected either from measurements or using a thermal simulator.
(2) The training data is used in a machine learning setup. Typically, such techniques can be summarized by Equation 43.

$$T_{pred} = \sum_{i=1}^{N} a_i x_i + b, \tag{43}$$

where x_i are the inputs or features of the algorithm, N is the number of samples used for training, and a and b are constants determined in the training phase. To obtain the transient temperature profile, generally the temperature at time instant t_n is used in estimating the temperature at the next time step t_{n+1}.

The objective behind training is to obtain the set of constants in Equation 43. This phase is generally quite long, often orders of magnitude higher than the runtime of the test phase.

(3) Once the constants are obtained, in the test phase, they are simply plugged into the generic equation to obtain the predicted values of temperature.

There are two primary categories of machine learning techniques employed in thermal simulation:

(1) **Feature extraction based techniques**:
Here a set of features are identified that have the potential to characterize the temperature profile. The constants in Equation 43 are identified and then used to predict the temperature profile for the test cases.
(2) **Neural network based techniques**:
Here a neural network is trained using a set of inputs that can characterize the temperature profile. Once the weights of the neural network are obtained, they are used in the test phase to predict the temperature profile quickly.

7.1 Feature Extraction based Techniques

Juan et al. [38] proposed an autoregressive model to train and predict the temperature profile. They used SPEC CPU2000 benchmarks as workloads, resulting in 100 different power configurations. For thermal simulation, they used the HotSpot tool. For power and performance simulation, Wattch [9] and SimpleScalar [10] were used respectively. The grid size was 32×32, resulting in 1024 grid points. 513 time frames, each of 0.1 ms were chosen. Hence, they trained their model on 513*100*1024 samples. The framework had two categories of features: physical features, and autoregressive (AR) features. The authors use five physical features [38]. These are: x location, y location, radius R, leakage power consumption and total power consumption. R is the distance of the grid point from the center. Since leakage power consumption has a higher temperature dependence, it was taken separately as a feature. The AR features comprise of the temperatures in adjoining grid cells and time frames. The authors chose 13 such AR features.

In the first step, k-means clustering was used to group the grids into two clusters, with each cluster having grids with similar total power consumption. The idea behind this was to partition thermal grids into high power consuming (power hungry) and moderately power consuming categories. The sum of squares within a cluster is minimized to determine the grids in each cluster. Once the clusters are obtained, the AR framework is applied separately to each cluster. To train the AR framework, the authors use Lasso regression. It is a regression analysis method that retains only the highly relevant features. The temperature of the i^{th} sample, T_i, is given by:

$$T_i = \sum_{j=1}^{P} \alpha_j m_{ij} + \beta, \tag{44}$$

where α and β are the coefficients to be determined, m_{ij} is a feature and P is the total number of features. In the training phase, the coefficients are learnt using Lasso regression. In the testing phase, the estimated coefficients are plugged into Equation 44 to predict the temperature profile. Although the training phase is long (takes 8 hours), in the testing phase the temperature profile can be quickly estimated by using the values of the predicted coefficients and the features. One of the features used for predicting the temperature at time $t+1$ is the temperature at previous time instant t. To obtain the prediction error, the authors use *10-fold cross validation*. In this method the sample is divided randomly into ten parts, and one part is used for validation. This is repeated ten times, ensuring each sample is used once for validation. The authors found that the root mean square error compared to HotSpot was 0.43°C for the power hungry cluster, and 0.2°C for the moderately powered cluster. In the online part, they achieved a 113X speedup over HotSpot. The authors also derived some insights regarding the nature of the features used for prediction. The coefficients of the radius R and leakage power were negative. It was concluded that high value of R corresponds

to grid points located on the rim of the chip, which has better heat dissipation properties. Similarly high leakage power means that the grid was idle, resulting in lower temperatures. The authors also found that the most important feature for predicting the temperature at time instant $t + 1$ was the temperature at the previous time instant t, which is intuitive since temperature values over time would be highly correlated until the input power values change.

Juan et al. [36, 37] used a similar approach to propose a variation aware thermal modeling framework. They used an autoregressive model to predict the maximum temperature of a layer, which is expressed as a combination of the leakage power consumption of each layer. The leakage current was found to be an exponential function of the effective channel length. The impact of the variation of effective channel length on leakage power was studied. 10-fold cross validation was used to determine the error. They compared their results using Hotspot as the standard. The prediction error rate for the maximum temperature was found to be 1.02%. They simply used the per layer leakage values for estimating maximum temperature, and did not integrate all the layers in the full chip. Based on these observations, they propose a stacking technique, which re-positions layers with high leakage, such that the maximum chip temperature is reduced.

Zhang et al. [97] propose a runtime system level thermal simulator and scheduler for HPC applications. They obtain a set of features from various sources and classify them into two categories: application features and physical features. Application features capture information about the characteristics of the workloads running on the system. Physical features carry information about the physical condition of the node, such as temperature sensor readings. They claim that using features from only the target node yields sufficient accuracy, and the improvement in accuracy by including its neighbors is minimal. The authors explore three prediction models: the Gaussian Process method proposed by them in which the physical feature vectors are assumed to have a joint Gaussian distribution; Lasso linear regression and multilayer perceptron. To reduce the set of features, they use the correlation feature selection algorithm. The average prediction errors obtained by them lies between 2.9 and 3.8°C , with time per prediction between 0.026 ms to 0.22 ms. They extend their model to propose a scheduler that schedules tasks such that the average temperature for the hottest node is minimized. However their technique is at the node level, and does not provide information about the temperature distribution within the node.

7.2 Neural Network based Techniques

Kumar and Atienza [41] proposed an on-chip neural network based thermal simulator that predicts the temperature profile for the next 500 ms. They used a linear neural network without an activation function, containing a single layer. They observe the similarity of Equation 43 with the state equation for a linear time invariant system:

$$x(t_{n+1}) = Ax(t_n) + Bi(t_n), \tag{45}$$

where $x(t)$ is a vector containing node potentials and $i(t)$ contains the source currents. Hence if A and B matrices are constants, which is the case when the thermal properties of the chip are constant, then a linear neural network is sufficient. Thus by giving $x(t_n)$ and $i(t_n)$ as the inputs, the output $x(t_{n+1})$ can be obtained. They train a single layer neural network to obtain the weights A and B. The weights A and B are considered to be of b bits each and are digitally stored by a circuit by adjusting the W/L ratio of the transistors to provide a binary-weighted current source array [41]. To reduce the complexity of the fully connected neural network, the authors propose to retain only the interconnections between neurons that are at a grid distance of r from each other. After the completion of each epoch, they round the weights to 2^{-b} to represent the digital quantization of weights. They used the fourth order Runge Kutta solver to generate the reference temperature profiles used for training. The authors claim that there are two main sources of inaccuracies. The

first source comes from ignoring the interactions between neurons at a distance greater than r. The second source of inaccuracy is the quantization of weights to b bits. They consider these two variables, r and b, to be design parameters. They were able to obtain a set of values for r and b such that the error at all points remained within 1-2 °C.

Sridhar et al. [77] used a similar approach to train their neural network. The reference input values were obtained from 3D ICE [78]. The RPROP [67] batch training algorithm was applied to speed up the training time of the neural network. To remove any floorplan dependence in the training process, they randomized the input power values. They implemented the online part of their algorithm on a GPU. For validation, they used a 2D IC with the UltraSPARC Niagara floorplan. With r equal to 3000 μm, they obtained an error less than 0.5°C . The neural network based simulator was found to be 35 times faster than 3D ICE at runtime, when implemented on a GPU.

Vincenzi et al. [84] also used a similar approach. They trained the neural network on a GPU. Each neuron was trained independently. The maximum error for a fully connected neural network was 0.25°C , which became slightly higher (0.35°C) with a loosely connected network with r = 5000 μm. Also, for GPU execution, the speed up compared to 3D ICE was up to 100X. Also, the GPU implementation was 6-9 times faster than a multi threaded CPU implementation.

Abad et al. [1] use feature selection in neural network based methods. They propose a thermal model for multicore processors in which fan speed, clock frequency, core temperature, and a few performance counters are used to train a multilayer perceptron. The authors define two types of features: behavioral and reflective. Behavioral parameters remain relatively constant over time and are not affected by the scheduler. Reflective parameters depend on the decisions of the scheduler. The authors use gradients of behavioral parameters as features, since their past values may govern their future behavior. For reflective parameters, their future gradients are taken as features. Next the authors apply feature selection to the set of features obtained. For this, the authors use a novel technique called minimal redundancy maximum relevance, which minimizes the number of features while simultaneously reducing the overlap between features. They use their thermal model to predict temperatures 2-5 prediction distances into the future. They choose 13 features using their feature selection method to predict temperatures for the next 5s and obtain a mean absolute error of 0.6 °C with a runtime per sample of about 3 μs.

8 DISCUSSION AND CONCLUSION

In this section, we discuss the characteristics of the techniques described in this paper. Additionally, we provide some insights into the discussed methods, and some key learnings. All results mentioned in this section have been obtained on an Intel i7-7700 Desktop computer running Ubuntu Linux 16.04.

8.1 Accuracy Needed at the Architectural Level

Since we describe early-stage chip level thermal estimation techniques in this paper, a primary problem to be addressed is to find the degree of accuracy needed at this level. We argue that a very high degree of accuracy is not required, since the inputs are highly variable. Two such sources of variability are: process variation and run-time variability.

8.1.1 Process Variation. Process variation is the variability that occurs when a device is fabricated. Due to uncertainties in the manufacturing process, the device parameters deviate from their expected values. For instance, the doping concentration in a fabricated device might vary at different locations. Similarly, the device dimensions and leakage current are impacted by process variation. All these lead to errors in temperature estimation. For instance let us assume that for a desired dopant

concentration of $10^{19}cm^{-3}$, the fabricated dopant concentration is obtained to be $10^{18}cm^{-3}$ [45]. This reduces the thermal conductivity of silicon [11] by roughly 10%. We simulated a 2D chip in HotSpot (for typical power profiles) with a corresponding reduction in thermal conductivity, and obtained the difference in maximum temperature to be 2.3%.

8.1.2 Runtime Variability. Runtime variability occurs due to a variety of reasons, the most common being non-determinism at the micro-architectural level introduced by multi-level caches, and on-chip networks. To estimate the effects of runtime variability, we ran a representative benchmark, *bzip2*, from the SPEC benchmark suite multiple times on an Intel i7 Desktop running Ubuntu Linux 16.04. The power variability was obtained to be 7% between different runs of the benchmark using Intel's RAPL (Running Average Power Limit) counter. Further, the variability in performance was obtained to be 3%. This results in an additional 6.6% variation in chip temperatures, as obtained using HotSpot 6.

Table 2. Variability in power, performance and temperature

Type of Variability	Amount of Variability	Temperature Variation
Dopant Fluctuation	$10X$	2.3%
Power	7%	6.6%

Such variations typically lead to additive errors. Hence, the temperature variation for a given design is of the order of 9-10% because of runtime factors. In the early stage design process, if our temperature estimation tool has an error limited to 3-4%, in our qualitative judgement it should be broadly acceptable if the speedup over traditional FEM simulation is several orders of magnitude.

8.2 Leakage Power

As discussed in this paper, very few techniques model leakage power analytically. The vast majority of thermal estimation techniques omit modeling leakage power entirely, which results in large errors. Since leakage power has an exponential dependence on temperature (theoretically), modeling the exact dependence is very complex and time consuming.

Now, considering that the accuracy requirement at the architectural level is not very high, we try to find an optimum leakage power model that provides acceptable accuracy, while being

Table 3. Leakage models commonly used

Model	**Works**
Linear	Liu et al.[53] Sarangi et al. [68] Sultan et al. [80]
Linearized around a reference	Wan et al. [85] Wang et al. [88] Yan et al. [91]
Exponential	Mesa-Martinez et al. [55] Singla et al. [72]
Higher Order Polynomial	Biswas et al. [7]

computationally inexpensive. For this purpose, we simulated a 7nm multifin FinFET and a 28nm UMC MOSFET using SPICE. Next, we used the BSIM equation to model leakage power within the standard operating range of real ICs (40°C to 80°C). Note that for small operating ranges, exponential functions can appear to be linear.

Model	**Error** (%)	**Time** (μs)
BSIM Equation	0	74.18
Linear fit	7.9	18.23
Quadratic fit	0.73	21.95
Piecewise linear	1.84	18.73

Table 4. Error and computation time for the BSIM leakage model and its approximations (40-80°C at increments of 0.1°C)

Table 5. Percentage error for the SPICE leakage model and its approximations

Model	**Error (%)**		
	7nm FinFET	28nm HVT CMOS	28nm LVT CMOS
Linear fit	7.3	9.7	4.0
Piecewise linear (3 segments)	1.7	2.2	0.9
Quadratic fit	0.6	1.15	0.2

Table 4 shows the computation time and percentage error while computing leakage power for 400 points using a linear leakage model, a quadratic model, a piecewise linear model and the complete BSIM equation, for temperature values between 40 and 80°C (increments of 0.1°C). Table 5 lists similar values obtained using SPICE as the baseline.

We found that using a simple linear leakage model results in an error of less than 8% in most cases. Since leakage power accounts for only 20-40% of the overall power, the net effect on temperature is reduced significantly. Assuming that 20% of the total power is leakage power, an 8% error in leakage power translates to a 1.33% error in temperature. A piecewise linear model results in an error of 0.33%. We conclude that a simple linear model provides an accuracy of over 98.6% for most cases, and may suffice in the early stages of the design process. To further improve the accuracy, a piecewise linear model may be used.

8.3 Tradeoffs of the Various Techniques

We have discussed four categories of techniques in this paper. Here we provide some insights based on our understanding into each of these categories, and when a particular method might be useful. We have simulated a representative $16 \times 16 \times 4$ 3D chip using a method from each of these categories. The leakage-converged results are summarized in Table 6.

8.3.1 Finite Element Method. The finite element method provides a high degree of accuracy. It can model any kind of chip in any type of environment and gives very accurate results. However the time needed to perform a typical simulation takes several minutes to a few hours, and even more for transient problems and 3D chips. Hence the finite element method is almost never used by reseachers in the architecture community to compute runtime temperature profiles. It is used

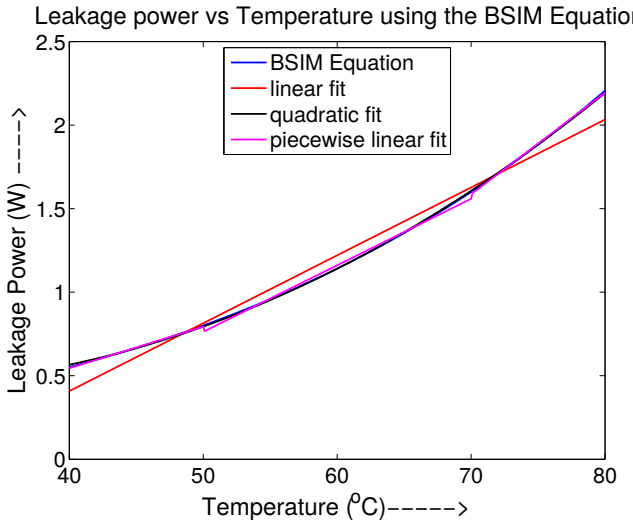

Fig. 8. Leakage power vs temperature using the BSIM equation

only for validation of new proposed techniques. We have chosen Ansys Icepak to simulate the representative chip using FEM.

8.3.2 Finite Difference Method. The finite difference method is widely used for all kinds of thermal simulations: computing the thermal profile of a basic 2D chip, modeling 3D chips, and transient thermal simulations with microchannels. Almost all open source simulators are based on the finite difference method. These methods offer a good trade off between speed and accuracy. We have used the 3DICE and HotSpot simulators in this category in our comparison study.

8.3.3 Green's Function based Methods. These set of techniques provide a significant speed up over finite difference methods with accuracies of the same order. Although capable of modeling all major thermal simulation problems, Green's function based methods have not received the attention of researchers as much as the finite difference methods. As a result, the ability to model fluid carrying microchannels is not present in this category of techniques till now. Open source simulators using this technique are also not in existence yet. However, the high online speed of the Green's function based method makes it the best suited for runtime applications where a high modeling speed is required. We have chosen 3DSim [80] as a representative simulator in this category.

8.3.4 Machine Learning based Methods. Machine learning has only recently been applied to solve thermal simulation problems. As a result, a very narrow set of problems have been solved using these methods. However, with the recent explosion in machine learning methods, we expect to see many more techniques emerge here. Since no open source simulators are available using machine learning methods, we have simulated a design similar to [41].

8.4 A Word on FinFETs and SoI Transistors

As shown by Allec et al. [3], the thermal effects of FinFETs at the chip level can be captured well using the traditional Fourier equation, with an error of less than 1%. The problems arise when we reach the device level, at dimensions below the mean free path of phonons ($\sim 200\ nm$). We have errors of 20-30% using simple Fourier equations. At this point, quantum effects come into play, and

Table 6. Representative techniques from each category and their timing

Technique	Steady State		Transient	
	Time	Accuracy	Time	Accuracy
Ansys Icepak	2 hours	-	12 hours	-
HotSpot	0.22s	3.4%	110s	7%
3DICE	1.67s	3.4%	215s	3.4%
3DSim	0.016s	5.5%	3.5s	6%
Machine Learning based	.075ms	3%	1.44 ms	10%

the Boltzmann transport equations have to be solved. Allec et al. propose an FEM based method to solve these equations.

Similarly, the self heating effects in SoI transistors have been studied well in the devices community, with solutions validated against the finite element method. However, the architectural level impact of substituting bulk silicon with SoI and its variants have not been studied in the context of thermal simulation.

8.5 Conclusion

Temperature estimation is a fundamental part of architectural design space exploration. This paper presents a comprehensive survey of the simulation techniques adopted by researchers to estimate the temperature profile in chips.

We believe that such a survey will help researchers to develop or improve the current simulation techniques to further reduce the error in thermal estimation. It should also enable researchers to choose the estimation method best suited for their applications. As the ICs get further scaled down, we look forward to newer and faster simulation techniques for 3D chips with microchannels and TSVs to emerge in the near future. We also expect more techniques addressing leakage power analytically to be proposed. We also expect to see more techniques based on machine learning methods to emerge in the near future. Also, parallel implementation of these techniques on GPUs, FPGAs, and manycore processors is necessary for further speedup.

REFERENCES

[1] Javad Mohebbi Najm Abad and Ali Soleimani. 2018. Novel Feature Selection Algorithm for Thermal Prediction Model. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 99 (2018), 1–14.

[2] Amir H Ajami, Massoud Pedram, and Kaustav Banerjee. 2001. Effects of non-uniform substrate temperature on the clock signal integrity in high performance designs. In *IEEE Conference on Custom Integrated Circuits*. IEEE, 233–236.

[3] Nicholas Allec, Zyad Hassan, Li Shang, Robert P Dick, and Ronggui Yang. 2008. ThermalScope: Multi-scale thermal analysis for nanometer-scale integrated circuits. In *International Conference on Computer-Aided Design*. IEEE Press, 603–610.

[4] Mohammad Asadzadeh. 2010. An Introduction to the Finite Element Method (FEM) for Differential Equations.

[5] Kaustav Banerjee, Massoud Pedram, and Amir H Ajami. 2001. Analysis and optimization of thermal issues in high-performance VLSI. In *Proceedings of the International Symposium on Physical Design*. ACM, 230–237.

[6] Theodore L Bergman and Frank P Incropera. 2011. *Introduction to heat transfer*. John Wiley & Sons.

[7] Susmit Biswas, Mohit Tiwari, Timothy Sherwood, Luke Theogarajan, and Frederic T Chong. 2011. Fighting fire with fire: modeling the datacenter-scale effects of targeted superlattice thermal management. In *International Symposium on Computer Architecture*. IEEE, 331–340.

[8] James R Black. 1969. Electromigration-A brief survey and some recent results. *IEEE Transactions on Electron Devices* 16, 4 (1969), 338–347.

[9] David Brooks, Vivek Tiwari, and Margaret Martonosi. 2000. *Wattch: A framework for architectural-level power analysis and optimizations*. Vol. 28. ACM.

[10] Doug Burger and Todd M Austin. 1997. The SimpleScalar tool set, version 2.0. *ACM SIGARCH computer architecture news* 25, 3 (1997), 13–25.
[11] Mihai G Burzo, Pavel L Komarov, and PE Raad. 2004. Non-contact thermal conductivity measurements of p-doped and n-doped gold covered natural and isotopically-pure silicon and their oxides. In *International Conference on Thermal and Mechanical Simulation and Experiments in Microelectronics and Microsystems. EuroSimE 2004.* IEEE, 269–276.
[12] Steven C Chapra and Raymond P Canale. 1998. *Numerical methods for engineers.* Vol. 2. McGraw-Hill New York.
[13] Yi-Kan Cheng, Prasun Raha, Chin-Chi Teng, Elyse Rosenbaum, and Sung-Mo Kang. 1998. ILLIADS-T: An electrothermal timing simulator for temperature-sensitive reliability diagnosis of CMOS VLSI chips. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 17, 8 (1998), 668–681.
[14] Jason Cong, Jie Wei, and Yan Zhang. 2004. A thermal-driven floorplanning algorithm for 3D ICs. In *International Conference on Computer Aided Design.* IEEE, 306–313.
[15] Ayse K Coskun, José L Ayala, David Atienza, and Tajana Simunic Rosing. 2009. Modeling and dynamic management of 3D multicore systems with liquid cooling. In *IFIP International Conference on Very Large Scale Integration (VLSI-SoC).* IEEE, 35–40.
[16] Ayse K Coskun, Jose L Ayala, David Atienza, Tajana Simunic Rosing, and Yusuf Leblebici. 2009. Dynamic thermal management in 3D multicore architectures. In *Design, Automation & Test in Europe.* IEEE, 1410–1415.
[17] J Crank and P Nicolson. 1947. A practical method for numerical evaluation of solutions of partial differential equations of the heat-conduction type. *Adv. Comput. Math* 6 (1947), 207–226.
[18] Shamik Das, Anantha Chandrakasan, and Rafael Reif. 2003. Three-dimensional integrated circuits: performance, design methodology, and CAD tools. In *IEEE Computer Society Annual Symposium on VLSI.* IEEE, 13–18.
[19] EG DâĂŹyakonov. 1964. Difference schemes of second-order accuracy with a splitting operator for parabolic equations without mixed derivatives. *Zh. Vychisl. Mat. I Mat. Fiz* 4 (1964), 935–941.
[20] Zhuo Feng and Peng Li. 2013. Fast thermal analysis on GPU for 3D ICs with integrated microchannel cooling. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 21, 8 (2013), 1526–1539.
[21] Alain Fourmigue, Giovanni Beltrame, and Gabriela Nicolescu. 2013. Explicit transient thermal simulation of liquid-cooled 3D ICs. In *Design, Automation & Test in Europe.* IEEE, 1385–1390.
[22] Alain Fourmigue, Giovanni Beltrame, and Gabriela Nicolescu. 2014. Efficient transient thermal simulation of 3D ICs with liquid-cooling and through silicon vias. In *Design, Automation & Test in Europe.* IEEE, 74.
[23] Alain Fourmigue, Giovanni Beltrame, Gabriela Nicolescu, and El Mostapha Aboulhamid. 2011. A linear-time approach for the transient thermal simulation of liquid-cooled 3D ICs. In *International Conference on Hardware/Software Codesign and System Synthesis (CODES+ ISSS),.* IEEE, 197–205.
[24] Kiyoshi Fukahori and Paul R Gray. 1976. Computer simulation of integrated circuits in the presence of electrothermal interaction. *IEEE Journal of Solid-State Circuits* 11, 6 (1976), 834–846.
[25] Yongkui Han and Israel Koren. 2007. Simulated annealing based temperature aware floorplanning. *Journal of Low Power Electronics* 3, 2 (2007), 141–155.
[26] Yongkui Han, Israel Koren, and C Mani Krishna. 2007. TILTS: A fast architectural-level transient thermal simulation method. *Journal of Low Power Electronics* 3, 1 (2007), 13–21.
[27] Yongkui Han, Israel Koren, and Csaba Andras Moritz. 2005. Temperature aware floorplanning. In *Workshop on Temperature Aware Computer Systems*, Vol. 24.
[28] Zyad Hassan, Nicholas Allec, Li Shang, Robert P Dick, Vishak Venkatraman, and Ronggui Yang. 2009. Multiscale thermal analysis for nanometer-scale integrated circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 28, 6 (2009), 860–873.
[29] Max A Heaslet and Robert F Warming. 1965. Radiative transport and wall temperature slip in an absorbing planar medium. *International Journal of Heat and Mass Transfer* 8, 7 (1965), 979–994.
[30] Wei Huang, Shougata Ghosh, Siva Velusamy, Karthik Sankaranarayanan, Kevin Skadron, and Mircea R Stan. 2006. HotSpot: A compact thermal modeling methodology for early-stage VLSI design. *IEEE Transactions on VLSI Systems* 14, 5 (2006), 501–513.
[31] Wei Huang, Kevin Skadron, Sudhanva Gurumurthi, Robert J Ribando, and Mircea R Stan. 2009. Differentiating the roles of IR measurement and simulation for power and temperature-aware design. In *IEEE International Symposium on Performance Analysis of Systems and Software.* IEEE, 1–10.
[32] Wei Hung, Charles Addo-Quaye, Theo Theocharides, Yuan Xie, N Vijakrishnan, and Mary Jane Irwin. 2004. Thermal-aware IP virtualization and placement for networks-on-chip architecture. In *International Conference on Computer Design: VLSI in Computers and Processors.* IEEE, 430–437.
[33] W-L Hung, Greg M Link, Yuan Xie, Narayanan Vijaykrishnan, and Mary Jane Irwin. 2006. Interconnect and thermal-aware floorplanning for 3D microprocessors. In *International Symposium on Quality Electronic Design.* IEEE Computer Society, 98–104.

[34] W-L Hung, Greg M Link, Yuan Xie, Narayanan Vijaykrishnan, and Mary Jane Irwin. 2006. Interconnect and thermal-aware floorplanning for 3D microprocessors. In *International Symposium on Quality Electronic Design*. IEEE, 6–pp.
[35] W-L Hung, Yuan Xie, Narayanan Vijaykrishnan, Charles Addo-Quaye, Theo Theocharides, and Mary Jane Irwin. 2005. Thermal-aware floorplanning using genetic algorithms. In *International Symposium on Quality of Electronic Design*. IEEE, 634–639.
[36] Da-Cheng Juan, Siddharth Garg, and Diana Marculescu. 2011. Statistical thermal evaluation and mitigation techniques for 3D chip-multiprocessors in the presence of process variations. In *Design, Automation & Test in Europe Conference & Exhibition*. IEEE, 1–6.
[37] Da-Cheng Juan, Siddharth Garg, and Diana Marculescu. 2014. Statistical peak temperature prediction and thermal yield improvement for 3D chip multiprocessors. *ACM Transactions on Design Automation of Electronic Systems* 19, 4 (2014), 39.
[38] Da-Cheng Juan, Huapeng Zhou, Diana Marculescu, and Xin Li. 2012. A learning-based autoregressive model for fast transient thermal analysis of chip-multiprocessors. In *Asia and South Pacific Design Automation Conference*. IEEE, 597–602.
[39] Yoon Jo Kim, Yogendra K Joshi, Andrei G Fedorov, Young-Joon Lee, and Sung-Kyu Lim. 2010. Thermal characterization of interlayer microfluidic cooling of three-dimensional integrated circuits with nonuniform heat flux. *Journal of Heat Transfer* 132, 4 (2010), 041009.
[40] Michael B Kleiner, Stefen A Kuhn, Peter Ramm, and Werner Weber. 1995. Thermal analysis of vertically integrated circuits. In *International Electron Devices Meeting*. IEEE, 487–490.
[41] Pratyush Kumar and David Atienza. 2010. Neural network based on-chip thermal simulator. In *International Symposium on Circuits and Systems*. IEEE, 1599–1602.
[42] Yuri A Kuznetsov. 2013. *Elements of applied bifurcation theory*. Vol. 112. Springer Science & Business Media.
[43] Clemens Lasance, Heinz Vinke, Harvey Rosten, and K-L Weiner. 1995. A novel approach for the thermal characterization of electronic parts. In *Semiconductor Thermal Measurement and Management Symposium*. IEEE, 1–9.
[44] Sang-Soo Lee and David J Allstot. 1993. Electrothermal simulation of integrated circuits: Analog and signal processing circuits. *IEEE Journal of Solid-State Circuits* 28, 12 (1993), 1283–1293.
[45] Greg Leung and Chi On Chui. 2012. Variability impact of random dopant fluctuation on nanoscale junctionless FinFETs. *IEEE Electron Device Letters* 33, 6 (2012), 767–769.
[46] Duo Li, Sheldon X-D Tan, Eduardo Hernandez Pacheco, and Murli Tirumala. 2009. Architecture-level thermal characterization for multicore microprocessors. *IEEE transactions on very large scale integration (VLSI) systems* 17, 10 (2009), 1495–1507.
[47] Duo Li, Sheldon X-D Tan, Eduardo H Pacheco, and Murli Tirumala. 2010. Parameterized architecture-level dynamic thermal models for multicore microprocessors. *ACM Transactions on Design Automation of Electronic Systems (TODAES)* 15, 2 (2010), 16.
[48] Peng Li, Lawrence T Pileggi, Mehdi Asheghi, and Rajit Chandra. 2004. Efficient full-chip thermal modeling and analysis. In *International conference on Computer-Aided Design*. IEEE Computer Society, 319–326.
[49] Ping-Chung Li and Tak K Young. 1996. Electromigration: the time bomb in deep-submicron ICs. *IEEE Spectrum* 33, 9 (1996), 75–78.
[50] W. Liu, K.M. Cao, X. Jin, and Chenming Hu. 2000. *BSIM 4.0.0 Technical Notes*. Technical Report UCB/ERL M00/39. EECS Department, University of California, Berkeley.
[51] Xue-Xin Liu, Zao Liu, Sheldon X-D Tan, and Joseph Gordon. 2012. Full-chip thermal analysis of 3D ICs with liquid cooling by GPU-accelerated GMRES method. In *International Symposium on Quality Electronic Design*. IEEE, 123–128.
[52] Xue-Xin Liu, Kuangya Zhai, Zao Liu, Kai He, Sheldon X-D Tan, and Wenjian Yu. 2015. Parallel thermal analysis of 3-D integrated circuits with liquid cooling on CPU-GPU platforms. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 23, 3 (2015), 575–579.
[53] Yongpan Liu, Robert P Dick, Li Shang, and Huazhong Yang. 2007. Accurate temperature-dependent integrated circuit leakage power estimation is easy. In *Design, Automation & Test in Europe*. IEEE, 1–6.
[54] ASME Majumdar. 1993. Microscale heat conduction in dielectric thin films. *Journal of Heat Transfer* 115, 1 (1993), 7–16.
[55] Francisco Javier Mesa-Martinez, Joseph Nayfach-Battilana, and Jose Renau. 2007. Power model validation through thermal measurements. *ACM SIGARCH Computer Architecture News* 35, 2 (2007), 302–311.
[56] P. K. Mittal. 2007. *Integral Transforms for Engineers and Physicists*. Har Anand Publishers.
[57] Hitoshi Mizunuma, Chia-Lin Yang, and Yi-Chang Lu. 2009. Thermal modeling for 3D-ICs with integrated microchannel cooling. In *International Conference on Computer-Aided Design*. ACM, 256–263.
[58] Joseph Nayfach-Battilana and Jose Renau. 2009. SOI, interconnect, package, and mainboard thermal characterization. In *International Symposium on Low power Electronics and Design*. ACM, 327–330.

[59] Yvan Notay. 2010. An aggregation-based algebraic multigrid method. *Electronic transactions on numerical analysis* 37, 6 (2010), 123–146.
[60] Dongkeun Oh, Charlie Chung Ping Chen, and Yu Hen Hu. 2007. 3DFFT: Thermal analysis of non-homogeneous IC using 3D FFT Green function method. In *International Symposium on Quality Electronic Design*. IEEE, 567–572.
[61] Je-Hyoung Park, Ali Shakouri, and Sung-Mo Kang. 2008. Fast evaluation method for transient hot spots in VLSI ICs in packages. In *International Symposium on Quality Electronic Design*. IEEE, 600–603.
[62] Massoud Pedram and Shahin Nazarian. 2006. Thermal modeling, analysis, and management in VLSI circuits: Principles and methods. *Proc. IEEE* 94, 8 (2006), 1487–1501.
[63] Kiran Puttaswamy and Gabriel H Loh. 2007. Thermal herding: Microarchitecture techniques for controlling hotspots in high-performance 3D-integrated processors. In *International Symposium on High Performance Computer Architecture*. IEEE, 193–204.
[64] Hanhua Qian, Hao Liang, Chip-Hong Chang, Wei Zhang, and Hao Yu. 2013. Thermal simulator of 3D-IC with modeling of anisotropic TSV conductance and microchannel entrance effects. In *Asia and South Pacific Design Automation Conference*. IEEE, 485–490.
[65] Haifeng Qian, Sachin S Sapatnekar, and Eren Kursun. 2012. Fast Poisson solvers for thermal analysis. *ACM Transactions on Design Automation of Electronic Systems (TODAES)* 17, 3 (2012), 32.
[66] Arifur Rahman and Rafael Reif. 2001. Thermal analysis of three-dimensional (3-D) integrated circuits (ICs). In *International Interconnect Technology Conference*. IEEE, 157–159.
[67] Martin Riedmiller and I Rprop. 1994. Rprop-description and implementation details. (1994).
[68] Smruti R Sarangi, Gayathri Ananthanarayanan, and Mahesh Balakrishnan. 2014. Lightsim: A leakage aware ultrafast temperature simulator. In *Asia and South Pacific Design Automation Conference*. IEEE, 855–860.
[69] Takashi Sato, Junji Ichimiya, Nobuto Ono, Koutaro Hachiya, and Masanori Hashimoto. 2005. On-chip thermal gradient analysis and temperature flattening for SoC design. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences* 88, 12 (2005), 3382–3389.
[70] Bing Shi and Ankur Srivastava. 2014. Optimized micro-channel design for stacked 3-D-ICs. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 33, 1 (2014), 90–100.
[71] Y-H Shih, Yusuf Leblebici, and S-M Kang. 1993. ILLIADS: A fast timing and reliability simulator for digital MOS circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 12, 9 (1993), 1387–1402.
[72] Gaurav Singla, Gurinderjit Kaur, Ali K Unver, and Umit Y Ogras. 2015. Predictive dynamic thermal and power management for heterogeneous mobile platforms. In *Design, Automation & Test in Europe*. IEEE, 960–965.
[73] Kevin Skadron, Mircea R Stan, Wei Huang, Sivakumar Velusamy, Karthik Sankaranarayanan, and David Tarjan. 2003. Temperature-aware microarchitecture. In *International Symposium on Computer Architecture*. IEEE, 2–13.
[74] E Hi Sondheimer. 1952. The mean free path of electrons in metals. *Advances in physics* 1, 1 (1952), 1–42.
[75] Shukri J Souri, Kaustav Banerjee, Amit Mehrotra, and Krishna C Saraswat. 2000. Multiple Si layer ICs: Motivation, performance analysis, and design implications. In *Design Automation Conference*. ACM, 213–220.
[76] Arvind Sridhar, Alessandro Vincenzi, David Atienza, and Thomas Brunschwiler. 2014. 3D-ICE: A compact thermal model for early-stage design of liquid-cooled ICs. *IEEE Trans. Comput.* 63, 10 (2014), 2576–2589.
[77] Arvind Sridhar, Alessandro Vincenzi, Martino Ruggiero, and David Atienza. 2012. Neural network-based thermal simulation of integrated circuits on GPUs. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 31, 1 (2012), 23–36.
[78] Arvind Sridhar, Alessandro Vincenzi, Martino Ruggiero, Thomas Brunschwiler, and David Atienza. 2010. 3D-ICE: Fast compact transient thermal modeling for 3D ICs with inter-tier liquid cooling. In *International Conference on Computer-Aided Design*. IEEE Press, 463–470.
[79] Haihua Su, Frank Liu, Anirudh Devgan, Emrah Acar, and Sani Nassif. 2003. Full chip leakage estimation considering power supply and temperature variations. In *International Symposium on Low Power Electronics and Design*. ACM, 78–83.
[80] Hameedah Sultan and Smruti R Sarangi. 2017. A fast leakage aware thermal simulator for 3D chips. In *Design, Automation & Test in Europe Conference & Exhibition*. IEEE, 1733–1738.
[81] Maxat N Touzelbaev, Josef Miler, Yizhang Yang, Gamal Refai-Ahmed, and Kenneth E Goodson. 2013. High-efficiency transient temperature calculations for applications in dynamic thermal management of electronic devices. *Journal of Electronic Packaging* 135, 3 (2013), 031001.
[82] Jeng-Liang Tsai, CC-P Chen, Guoqiang Chen, Brent Goplen, Haifeng Qian, Yong Zhan, Sung-Mo Kang, Martin DF Wong, and Sachin S Sapatnekar. 2006. Temperature-aware placement for SOCs. *Proc. IEEE* 94, 8 (2006), 1502–1518.
[83] Dragica Vasileska, Katerina Raleva, and Steve M Goodnick. 2010. Electrothermal studies of FD SOI devices that utilize a new theoretical model for the temperature and thickness dependence of the thermal conductivity. *IEEE Transactions on Electron Devices* 57, 3 (2010), 726–728.

[84] Alessandro Vincenzi, Arvind Sridhar, Martino Ruggiero, and David Atienza. 2011. Fast thermal simulation of 2D/3D integrated circuits exploiting neural networks and GPUs. In *International Symposium on Low-power Electronics and Design*. IEEE Press, 151–156.

[85] Jiachun Wan, Hai Wang, Sheldon X-D Tan, Chi Zhang, Yuan Yuan, Keheng Huang, and Zhenghong Zhang. 2016. A fast full-chip static power estimation method. In *International Conference on Solid-State and Integrated Circuit Technology*. IEEE, 241–243.

[86] Baohua Wang and Pinaki Mazumder. 2007. Accelerated chip-level thermal analysis using multilayer Green's function. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 26, 2 (2007), 325–344.

[87] Hai Wang, Jiachun Wan, Sheldon Tan, Chi Zhang, He Tang, Yuan Yuan, Keheng Huang, and Zhenghong Zhang. 2018. A Fast Leakage-Aware Full-Chip Transient Thermal Estimation Method. *IEEE Trans. Comput.* (2018).

[88] Hai Wang, Jiachun Wan, Sheldon X-D Tan, Chi Zhang, He Tang, Yuan Yuan, Keheng Huang, and Zhenghong Zhang. 2018. A Fast Leakage-Aware Full-Chip Transient Thermal Estimation Method. *IEEE Trans. Comput.* 67, 5 (2018), 617–630.

[89] Ting-Yuan Wang, Yu-Min Lee, and Charlie Chung-Ping Chen. 2003. 3D thermal-ADI: an efficient chip-level transient thermal simulator. In *International Symposium on Physical design*. ACM, 10–17.

[90] Chao Yan, Hengliang Zhu, Dian Zhou, and Xuan Zeng. 2017. An efficient leakage-aware thermal simulation approach for 3D-ICs using corrected linearized model and algebraic multigrid. In *Design, Automation & Test in Europe*. IEEE, 1207–1212.

[91] Chao Yan, Hengliang Zhu, Dian Zhou, and Xuan Zeng. 2017. An efficient leakage-aware thermal simulation approach for 3D-ICs using corrected linearized model and algebraic multigrid. In *Design, Automation & Test in Europe*. IEEE, 1207–1212.

[92] Yonghong Yang, Zhenyu Gu, Changyun Zhu, Robert P Dick, and Li Shang. 2007. ISAC: Integrated space-and-time-adaptive chip-package thermal analysis. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 26, 1 (2007), 86–99.

[93] Yong Zhan, Sanjay V Kumar, Sachin S Sapatnekar, et al. 2008. Thermally aware design. *Foundations and Trends® in Electronic Design Automation* 2, 3 (2008), 255–370.

[94] Yong Zhan and Sachin S Sapatnekar. 2005. Fast computation of the temperature distribution in VLSI chips using the discrete cosine transform and table look-up. In *Asia and South Pacific Design Automation Conference*. ACM, 87–92.

[95] Yong Zhan and Sachin S Sapatnekar. 2005. A high efficiency full-chip thermal simulation algorithm. In *International Conference on Computer-Aided Design*. IEEE Computer Society, 635–638.

[96] Yong Zhan and Sachin S Sapatnekar. 2007. High-efficiency Green function-based thermal simulation algorithms. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 26, 9 (2007), 1661–1675.

[97] Kaicheng Zhang, Akhil Guliani, Seda Ogrenci-Memik, Gokhan Memik, Kazutomo Yoshii, Rajesh Sankaran, and Pete Beckman. 2018. Machine Learning-Based Temperature Prediction for Runtime Thermal Management Across System Components. *IEEE Transactions on Parallel and Distributed Systems* 29, 2 (2018), 405–419.

[98] Runjie Zhang, Mircea R Stan, and Kevin Skadron. Aug 26, 2015. *HotSpot 6.0: Validation, Acceleration and Extension*. Technical Report CS-2015-04. University of Virginia, Charlottesville, VA.

[99] Pingqiang Zhou, Yuchun Ma, Zhouyuan Li, Robert P Dick, Li Shang, Hai Zhou, Xianlong Hong, and Qiang Zhou. 2007. 3D-STAF: scalable temperature and leakage aware floorplanning for three-dimensional integrated circuits. In *International Conference on Computer-Aided Design*. IEEE, 590–597.

[100] Xiuyi Zhou, Yi Xu, Yu Du, Youtao Zhang, and Jun Yang. 2008. Thermal management for 3D processors via task scheduling. In *International Conference on Parallel Processing*. IEEE, 115–122.

[101] Amirkoushyar Ziabari, Je-Hyoung Park, Ehsan K Ardestani, Jose Renau, Sung-Mo Kang, and Ali Shakouri. 2014. Power blurring: Fast static and transient thermal analysis method for packaged integrated circuits and power devices. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 22, 11 (2014), 2366–2379.

[102] Amir Zjajo, Nick Van Der Meijs, and Rene Van Leuken. 2012. Thermal analysis of 3D integrated circuits based on discontinuous Galerkin finite element method. In *International Symposium on Quality Electronic Design*. IEEE, 117–222.