LightSim : A Leakage Aware Ultrafast Temperature Simulator

Smruti R. Sarangi
Computer Science and Engg.
Indian Institute of Technology
Hauz Khas, New Delhi, 110016
Tel: +91-11-2659 7065
e-mail: srsarangi@cse.iitd.ac.in

Gayathri Ananthanarayanan
School of Information Technology
Indian Institute of Technology
Hauz Khas, New Delhi, 110016
Tel: +91-11-2659 6041
e-mail: gayathri@cse.iitd.ac.in

M. Balakrishnan
Computer Science and Engg.
Indian Institute of Technology
Hauz Khas, New Delhi, 110016
Tel: +91-11-2659 1285
e-mail: mbala@cse.iitd.ac.in

Abstract— In this paper, we propose the design of an ultra-fast temperature simulator (LightSim) that can perform both steady state and transient thermal analysis, and also take the effect of leakage power into account. We use a novel Hankel transform based technique to derive a transient version of the Green's function for a chip, which takes into account the feedback loop between temperature and leakage. Subsequently, we calculate the temperature map of a chip by convolving the derived Green's function with the power map. Our simulator is at least 3500 times faster than HotSpot, and at least 2.3 times faster than competing research prototypes [4, 12]. The total error is limited to 0.18 $^\circ$C .

I. Introduction

Since the last ten years, on-chip temperature is increasingly being regarded as a first class design constraint. Temperature has a direct effect on the amount of leakage power, and lifetime reliability. Hence, it is necessary for designers to get estimates of chip temperature at an early stage of the design process. In response to this requirement researchers in both industry and academia have proposed a wide variety of simulation tools for estimating on-chip temperature. The existing simulation tools primarily solve two variants of thermal problems – estimating steady state temperatures [1. `steady state` problem], and estimating transient temperatures [2. `transient` problem]. If the feedback loop between temperature and leakage is considered, then we have two more problems namely the [3. `steady state leakage` problem], and the [4. `transient leakage` problem].

Researchers have mostly built on generic methodologies to model temperature such as the finite difference and finite element methods. To take the leakage feedback loop into account most tools run the temperature simulation several times till the temperature and leakage power values converge. Unfortunately, most of the finite element and finite difference based methods are fairly slow. Some of the fastest tools developed over the last decade such as HotSpot 6 [14], and SescTherm [19] take several minutes to solve the `transient` and `transient leakage` problems. Hence, researchers have proposed to use novel Green's function based methods [18, 17] that have been shown to be 2-3X faster [19] than finite element/difference methods. Their main drawback is that they are not suitable for transient thermal analysis.

In this paper, we propose the design of LightSim, an ultrafast thermal simulator that solves all the four thermal problems in less than 13 ms on a standard Intel i7 desktop processor. It is primarily designed for estimating the on-chip temperature of multicore processors. As pointed out in prior work [14, 4], the aim of designing an architecture level temperature estimator is to broadly estimate the high level thermal profile, and thus an extremely high level of accuracy is not necessary. Hence, we make a set of simplistic assumptions to speed up our simulation time (similar to [4, 12, 7]). LightSim is at least 3500 times faster than the most popular open source simulator, HotSpot, and is about 2-4 times faster than the fastest research prototypes – CONTILTS [4] and PowerBlur [12].

The crux of our technique is to create a transient version of the Green's function (impulse response of a point power source) that takes the leakage feedback loop into account. We use a novel Hankel transform based technique to arrive at this approximate version of the Green's function. Subsequently, we derive the thermal profile using a standard method [18, 17], which convolves the derived Green's function with the power profile. Our approach effectively combines the positive aspects of Green's function based methods (speed) with that of traditional finite difference based methods (incorporation of transient, and leakage effects). Our approach does have some deficiencies though. At the moment, the leakage values at the rim of the chip are not derived rigorously, and use empirical factors. Even though the error in estimation is small, we intend to propose a more rigorous framework in the future.

II. Background and Related Work

A. Source of Power Dissipation

There are two sources of power dissipation in a processor – dynamic and static(leakage) power. Dynamic power is dissipated because of the switching activity in circuits. It is independent of temperature. It is dependent on the workload, and the micro-architecture.

However, the static or leakage power is a function of temperature, and is traditionally described by Equation 1 in the simplified BSIM 4 [8] model.

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

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

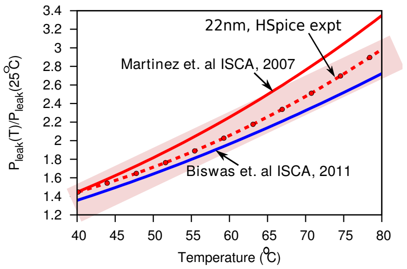

Fig. 1. Leakage models

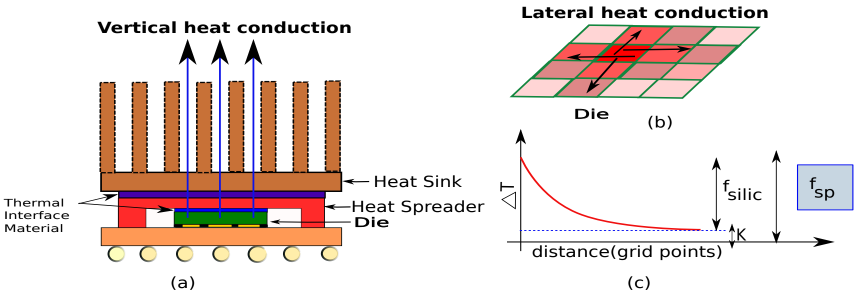

Fig. 2. (a) Processor package (b) Lateral heat conduction (c) f_{sp} and f_{silic}

P_{leak} has a complex dependence on temperature, and is exponentially dependent on temperature for high values($> 500mV$) of the threshold voltage (V_{th}). In modern processors, the threshold voltages are about 150 mV, and the relationship between leakage power and temperature becomes approximately linear (based on Equation 1). Two highly cited recent papers ([10] and [1]) have also arrived at the same conclusion using accurate physical measurements and HSpice simulations. We also conducted HSpice simulations for finding the dependence of leakage on temperature for the 22nm technology node, and the results are shown in Figure 1, along with the results of [10, 1]. The operating range of a processor is typically between 45°C and 70°C . In this range, leakage is approximately linear as we can see in Figure 1. This fact has been used to speed up thermal simulators [9], and dynamic thermal management algorithms [5, 6]. All the authors have reported less than 5% error with a linear model of leakage. Note that to estimate temperature, we need to repeatedly estimate leakage, and then the resultant temperature, till convergence.

B. Lateral Heat Conduction

Figure 2(a) shows a diagram of the processor's package. The processor die generates heat because of dynamic and leakage power dissipation. There is some amount of lateral heat conduction as shown in Figure 2(b); however, a larger fraction of the heat escapes through the heat spreader and sink to the ambient (environment). The heat spreader is a copper plate that helps to homogenize the temperature distribution of the silicon die.

Now, if we apply a point heat source at the center of the die, then the temperature distribution is mostly isotropic (independent of direction), and the relation between ΔT, and the radial distance (measured in grid points) is conceptually shown in Figure 2(c). We observe that the heat spread function f_{sp} can be divided into two parts. The first part is a rapidly decaying function (f_{silic}) that captures the temperature rise in the adjoining grid points. It represents the fact that lateral heat conduction is limited to a small region because of the conductivity of silicon (140 W/m-K) is lower than the conductivity of copper (400 W/m-K). The second part is a constant κ that captures the effect of the entire die heating up because of a point power source. This happens because some of the heat is transferred to the spreader, which in turn transfers some energy to all the points in the die. This heat spread function (f_{sp}) is also known as the Green's function of the system at the center of the die. We thus have:

$$f_{sp} = f_{silic} + \kappa \tag{2}$$

By conducting empirical measurements using the Hotspot simulator, we concluded that the radially symmetric f_{sp} function accurately describes the temperature rise for most of the points on the die other than the rim (around 10% of the total area). This is because f_{silic} decays very quickly.

C. Related Work on Temperature Simulation

All the temperature simulation algorithms essentially solve the Fourier's heat equation:

$$\frac{\partial \mathcal{U}}{\partial t} = \alpha_1 \nabla^2 \mathcal{U} + \alpha_2 Q \tag{3}$$

Here, $\mathcal{U}(x, y, z, t)$ is the temperature field, ∇ is the Laplacian operator, Q is the power profile, α_1, and α_2 are constants. The boundary conditions typically specify adiabatic (no heat flow) side/bottom boundaries, and a constant temperature above the die.

The most commonly used method to solve Fourier's equation is the finite difference method. The finite difference method models the points in the package as a grid, and then for each grid point it transforms the Fourier equation into a linear recurrence relation. For example it transforms $\partial\mathcal{U}(x, y, z, t)/\partial t$ to $(\mathcal{U}(x, y, z, t + h) - \mathcal{U}(x, y, z, t))/h$ for grid point (x, y, z). We can solve the resulting set of recurrence relations using standard techniques in linear algebra. The accuracy of this method is a function of the number of grid points, n. Since matrix multiplication, and inversion are essentially $O(n^3)$ operations, this method is very slow. It can be accelerated by the alternating direction implicit method [15], model order reduction using Krylov subspaces [16], and multigrid methods [16]. The popular tool, Hotspot [14], creates an electrical circuit based on the recurrence equations, and processes it using existing circuit simulation methods. Similarly, the CONTILTS [4] simulator significantly speeds up the simulation by using a piece wise constant approximation for the variation of power against time, and by using results from the theory of linear systems. Researchers have experimented with the slower finite element method [3]. However, architectural thermal simulations do not need such high level of accuracy, at the cost of simulation speed.

A recent set of approaches [18, 17, 12] compute the 2D impulse response (Green's function, G) of an unit power source in the center of the die, and compute the change in the temperature field ($\mathcal{U}$) as:

$$\mathcal{U} = G \star Q \tag{4}$$

Here, $\star$ is the 2 dimensional convolution operator, and Q is the power map. The Green's function is essentially f_{sp} in 2D Cartesian co-ordinates for most of the die (see Section B). We wish to create a time varying version of the Green's function that takes leakage power into account, and uses Equation 4 for the `transient` and `transient leakage` problems.

III. Derivation of the Green's Functions

Symbol	Full Form	Meaning
$\mathcal{U}$	$\mathcal{U}(r,t)$	Temperature field
P_{dyn}	$P_{dyn}(x,y)$	Dynamic power
P_{leak}	$P_{leak}(x,y)$	Leakage power
β		dP_{leak}/dT
f_{sp}	$f_{sp}(r,t)$	Steady state Green's function (polar co-ordinates)
f_{silic}	$f_{silic}(r,t)$	Local heat spread function (polar co-ordinates)
κ		Global heat spread ($f_{sp} - f_{silic}$)
κ_1		$\beta\kappa \int\int_{\mathcal{A}} f_{sp} dx.dy$
κ_2		$2\pi\beta(\kappa+\kappa_1)(\mathcal{H}(f_{silic}) \mid_{s=0})$
ϕ		$\kappa+\kappa_1+\kappa_2$
f_{leaksp}	$f_{leaksp}(r,t)$	Leakage aware Green's function (polar co-ordinates)
f_α	$f_\alpha(s)$	$2\pi C(1+2\pi\beta\mathcal{H}(f_{silic}))$
f_{inv}	$f_{inv}(r,t)$	$\mathcal{H}^{-1}(\mathcal{H}(f_{leaksp})\times$ $e^{-\frac{t}{f_\alpha(s)(\kappa\delta(s)/s+\mathcal{H}(f_{silic}))}})$
		$f_{inv} = f_{inv}{}_0^\epsilon + f_{inv}{}_\epsilon^\infty$
f_{trans}	$f_{trans}(r,t)$	Transient Green's function with the effect of leakage
Functions: $\mathcal{U}$,f_{sp}, f_{silic} are also used in Cartesian co-ordinates		

TABLE I
GLOSSARY

A. `Steady State Leakage` Problem

Let us start from Equation 4, and use $f_{sp}(x,y)$ as the Green's function without considering the effects of leakage. Secondly, let us split the power into two parts – P_{dyn} (dynamic), and P_{leak} (leakage). Let $\mathcal{U}_0$ be the temperature field with no dynamic power, and $\mathcal{U}_P$ denote the final temperature field. Let P_{leak0} be the leakage field at $\mathcal{U}_0$. We thus have:

$$\begin{aligned}\mathcal{U} = \mathcal{U}_P - \mathcal{U}_0 &= f_{sp} \star (P_{dyn} + P_{leak}) - f_{sp} \star P_{leak0} \\ &= f_{sp} \star (P_{dyn} + \Delta P_{leak})\end{aligned} \tag{5}$$

Let us now assume P_{dyn} to be the Dirac delta function, δ (a point source with total power of 1 Watt). Hence, $f_{sp} \star P_{dyn} = f_{sp}$. Secondly, since we approximate leakage with a linear model. We have $\Delta P_{leak} = \beta\mathcal{U}$ (β is a constant of proportionality). Thus:

$$\mathcal{U} = f_{sp} + \beta f_{sp} \star \mathcal{U} \tag{6}$$

Using Equation 2 ($f_{sp} = f_{silic} + \kappa$) we get:

$$\begin{aligned}\mathcal{U} &= f_{silic} + \kappa + \beta f_{silic} \star \mathcal{U} + \beta(\kappa \star \mathcal{U}) \\ &= f_{silic} + \kappa + \beta f_{silic} \star \mathcal{U} + \beta\kappa \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \mathcal{U} dx.dy \\ &\approx f_{silic} + \kappa + \beta f_{silic} \star \mathcal{U} + \underbrace{\beta\kappa \int\int_{\mathcal{A}} f_{sp} dx.dy}_{\kappa_1} \\ &= f_{silic} + \kappa + \beta f_{silic} \star \mathcal{U} + \kappa_1\end{aligned} \tag{7}$$

We make a simplifying assumption here by assuming that the integral of $\mathcal{U}$ over the area of the chip ($\mathcal{A}$) is approximately equal to the integral of f_{sp} over $\mathcal{A}$. Let us now compute 2D Fourier transforms of the LHS and RHS. The 2D Fourier transform is defined as:

$$\mathcal{F}(s,t) = \frac{1}{2\pi}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} e^{-j(xs+yt)} dxdy \tag{8}$$

Note that the Fourier transform (as defined by Equation 8) of a convolution is equal to the product of Fourier transforms multiplied by 2π. We thus have from Equation 7:

$$\mathcal{F}(\mathcal{U}) = \mathcal{F}(f_{silic}) + (\kappa+\kappa_1)\mathcal{F}(1) + 2\pi\beta\mathcal{F}(f_{silic})\mathcal{F}(\mathcal{U}) \tag{9}$$

Now, f_{silic} is radially symmetric because we are assuming a large (theoretically infinite) die size. As mentioned in Section B, this assumption holds for about 90% of the area of a typical 2x2 cm^2 die. Let us now use zero order Hankel transforms($\mathcal{H}$) [11] to reduce the 2D problem to a 1D problem. A zero order Hankel transform is equivalent to 2D Fourier transforms of radially symmetric functions in polar coordinates [11]. A Hankel transform is defined as:

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

Here $\mathcal{J}_0$ is a Bessel function of the first kind of order 0. The inverse Hankel transform has exactly the same form with r replaced by the transform variable s. The Hankel transform of a 2D convolution is the product of the transforms multiplied by 2π. From Equation 9, we have after converting to polar coordinates.

$$\begin{aligned}&\mathcal{H}(\mathcal{U}) = \mathcal{H}(f_{silic}) + (\kappa+\kappa_1)\mathcal{H}(1) + 2\pi\beta\mathcal{H}(f_{silic})\mathcal{H}(\mathcal{U}) \\ &\Rightarrow \mathcal{H}(\mathcal{U}) = \frac{\mathcal{H}(f_{silic}) + (\kappa+\kappa_1)\mathcal{H}(1)}{1 - 2\pi\beta\mathcal{H}(f_{silic})}\end{aligned} \tag{11}$$

β is typically a small value (0.09 to 0.006), and the peak value of $\mathcal{H}(f_{silic})$ is limited to small values($\approx$ 1) as observed in our simulations. Thus, we can simplify Equation 11 by considering the fact that $(1-x)^{-1} \approx 1+x$, when $x \ll 1$. We use the result: $\mathcal{H}(f(r) \star g(r)) = 2\pi\mathcal{H}(f(r)\mathcal{H}(g(r))$.

$$\begin{aligned}\mathcal{H}(\mathcal{U}) &= (\mathcal{H}(f_{silic}) + (\kappa+\kappa_1)\mathcal{H}(1)) \times (1 + 2\pi\beta\mathcal{H}(f_{silic})) \\ &= \mathcal{H}(f_{silic}) + 2\pi\beta\mathcal{H}(f_{silic})^2 + \mathcal{H}(\kappa+\kappa_1) \\ &\quad + 2\pi\beta(\kappa+\kappa_1)\mathcal{H}(1)\mathcal{H}(fsilic)\end{aligned} \tag{12}$$

$\mathcal{H}(1)$ is $\delta(s)/s$, and is thus 0 everywhere other than 0. Hence, $\mathcal{H}(1)\mathcal{H}(f_{silic})$ = $\mathcal{H}(1)$ $(\mathcal{H}(f_{silic}) \mid_{s=0})$. Thus:

$$\begin{aligned}\mathcal{U} &= f_{silic} + 2\pi\beta\mathcal{H}^{-1}(\mathcal{H}(f_{silic})^2) + \kappa + \kappa_1 \\ &\quad + \underbrace{2\pi\beta(\kappa+\kappa_1)(\mathcal{H}(f_{silic}) \mid_{s=0})}_{\kappa_2} \\ &= f_{silic} + 2\pi\beta\mathcal{H}^{-1}(\mathcal{H}(f_{silic})^2) + \underbrace{(\kappa+\kappa_1+\kappa_2)}_{\phi}\end{aligned} \tag{13}$$

$$\boxed{f_{leaksp} = f_{silic} + 2\pi\beta\mathcal{H}^{-1}(\mathcal{H}(f_{silic})^2) + \phi}$$

Here $\mathcal{U}$ (referred to as f_{leaksp}) is the modified Green's function that takes into account the effect of leakage.

B. `Transient Leakage` Problem

If we consider the transient case, the resultant temperature field is defined by: (refer to [14]).

$$\mathcal{U} = f_{sp} + \beta f_{sp} \star \mathcal{U} - C f_{sp} \star \frac{\partial\mathcal{U}}{\partial t} \tag{14}$$

Here, C is a constant (thermal capacitance). By following the same set of steps that we followed to derive Equation 12, and the Leibnitz rule, we get:

$$\begin{aligned}\mathcal{H}(\mathcal{U}) &= \mathcal{H}(f_{leaksp}) - \\ &\quad \underbrace{2\pi C(1+2\pi\beta\mathcal{H}(f_{silic}))}_{f_\alpha} \times \mathcal{H}(f_{sp})\mathcal{H}\left(\frac{\partial\mathcal{U}}{\partial t}\right) \\ &= \mathcal{H}(f_{leaksp}) - f_\alpha\mathcal{H}(f_{sp})\mathcal{H}\left(\frac{\partial\mathcal{U}}{\partial t}\right) \\ &= \mathcal{H}(f_{leaksp}) - f_\alpha\mathcal{H}(\kappa + f_{silic})\frac{\partial\mathcal{H}(\mathcal{U})}{\partial t}\end{aligned} \tag{15}$$

Solving for t, and applying the boundary conditions: $\mathcal{H}(\mathcal{U}) = 0 \mid_{t=0}$, and $\mathcal{H}(\mathcal{U}) = \mathcal{H}(f_{leaksp}) \mid_{t=\infty}$.

$$\begin{aligned}&\mathcal{H}(\mathcal{U}) = \mathcal{H}(f_{leaksp}) \times \left(1 - e^{-\frac{t}{f_\alpha\mathcal{H}(\kappa+f_{silic})}}\right) \\ &\Rightarrow \mathcal{H}(\mathcal{U}) = \mathcal{H}(f_{leaksp}) \times (1 - e^{-\frac{t}{f_\alpha(s)(\kappa\delta(s)/s+\mathcal{H}(f_{silic}))}})\end{aligned} \tag{16}$$

The Hankel transform of κ is $\kappa\delta(s)/s$. It is ∞ for $s = 0$, and is 0 for all other values of s. Let us now evaluate the inverse Hankel transform (f_{inv}) of $\mathcal{H}(f_{leaksp}) \times e^{-\frac{t}{f_\alpha(s)(\kappa\delta(s)/s+\mathcal{H}(f_{silic}))}}$.

Let us approximate $\delta(s)$ as a function that is equal to $1/\epsilon$ from 0 to ϵ, and is 0 everywhere else. Here, $\epsilon \to 0$. We can thus break the inverse Hankel transform f_{inv} into two parts – ${f_{inv}}_0^{\epsilon}$, and ${f_{inv}}_{\epsilon}^{\infty}$. ${f_{inv}}_{\epsilon}^{\infty}$ can be calculated by using the formula for the inverse Hankel transform, and numerical integration.

$$\begin{aligned}{f_{inv}}_{\epsilon}^{\infty}(r,t) = \int_{\epsilon}^{\infty} &\left(\mathcal{H}(f_{silic}) + 2\pi\beta\mathcal{H}(f_{silic})^2\right) \times \\ &e^{-\frac{t}{2\pi C(1+2\pi\beta\mathcal{H}(f_{silic}))(\mathcal{H}(f_{silic}))}}\mathcal{J}_0(sr)sds\end{aligned} \tag{17}$$

Let us now compute ${f_{inv}}_0^{\epsilon}$. Since $\delta(s)/s \gg \mathcal{H}(f_{silic})$ when $s < \epsilon$, we can ignore all the terms with $\mathcal{H}(f_{silic})$. We can thus approximate $\mathcal{H}(f_{leaksp}) = \phi\delta(s)/s$ (Equation 13). We thus have:

$$\begin{aligned}{f_{inv}}_0^{\epsilon}(r,t) &= \int_0^{\epsilon} (\phi\delta(s)/s) \times e^{-\frac{t}{f_\alpha(s)(\kappa\delta(s)/s)}} s\mathcal{J}(sr)ds \\ &= \int_0^{\epsilon} (\phi/\epsilon) \times e^{-\frac{ts\epsilon}{f_\alpha(s)\kappa}}\mathcal{J}(sr)ds \qquad (\delta(s) = 1/\epsilon)\end{aligned} \tag{18}$$

Since $\epsilon \to 0$, the product $sr \to 0$, and thus $\mathcal{J}_0(sr) \to 1$. Secondly, f_α tends to $(f_{\alpha 0} = 2\pi C(1 + 2\pi\beta\mathcal{H}(f_{silic})(0))$, because s tends to 0. By making these substitutions, we get:

$$\begin{aligned}{f_{inv}}_0^{\epsilon}(r,t) &= \int_0^{\epsilon} (\phi/\epsilon) \times e^{-\frac{ts\epsilon}{f_{\alpha 0}\kappa}}ds \\ &= \frac{\phi f_{\alpha 0}\kappa}{t\epsilon^2}\left(1 - e^{-\frac{t\epsilon^2}{f_{\alpha 0}\kappa}}\right)\end{aligned} \tag{19}$$

Note that when t is small, ${f_{inv}}_0^{\epsilon}$ tends to ϕ. It becomes zero as $t \to \infty$. To conclude, the Green's function, f_{trans}, for the `transient leakage` problem is:

$$\boxed{f_{trans}(r,t) = f_{leaksp}(r) - {f_{inv}}_0^{\epsilon}(r,t) - {f_{inv}}_{\epsilon}^{\infty}(r,t)} \tag{20}$$

${f_{inv}}_0^{\epsilon}$ incorporates the effect of heating up of the entire package. This is a slowly increasing function. ${f_{inv}}_{\epsilon}^{\infty}$ is a much faster growing function and models the transient temperature rise in the neighborhood of the power source. Lastly, note that we assume that the leakage power increases instantaneously (quasi-static assumption). Thermal time scales are at least of the order of microseconds, and this is too large for non quasi static effects to set in [8, 2].

C. Corrections for the Edges and Corners

We use the technique proposed by Park et. al. [12] to make corrections for the rim of the chip. Park et. al. use three different Green's functions corresponding to the center, edge, and corner of the chip. We do the same and use the appropriate version of f_{sp} in Equations 6, and 14. We need to also make a correction to the term $\beta f_{sp} \star \mathcal{U}$ in Equations 6 and 14. This term represents the feedback component in the temperature leakage loop. A primary power source increases the temperature in the neighborhood, and each point in the neighborhood starts acting as a secondary power source dissipating leakage power. Since the size of the neighborhood is reduced to a quarter for the corners, and reduced to half for the edges, we divide the term $\beta f_{sp} \star \mathcal{U}$ by 4 for the corners, and 2 for the edges. These two corrections help us significantly reduce the error in temperature estimation for the rim of the chip.

D. Using the Green's Functions

This part of the algorithm is the same as prior approaches that use Green's functions [18, 17]. They compute the 2D convolution of the Green's function and the power map. This is typically done by computing the Fourier transform of both the functions, multiplying them, and then computing the inverse transform. We use Equations 12, and 16, to get the 2D Fourier transform of the Green's function, and then proceed to obtain the temperature map. We accelerate the process of temperature estimation by pre-computing the Green's function, and using a fast lookup table based approach similar to [17].

IV. Results

A. Setup

We implemented LightSim in two parts. The offline part is written in R [13], and the online part in C. The offline part first invokes the popular Hotspot [14] tool to compute three different Green's functions (center, edge, corner) for a chip with the parameters shown in Table II.

Parameter	Value
Die size	400 mm^2
Silicon conductivity Spreader conductivity	130 W/m-K 370 W/m-K
Heatsink conductivity	237 W/m-K
Convection resistance	0.1 K/W
Spreader thickness	3.5 mm
Heatsink thickness	24.9 mm

TABLE II
Hotspot configuration

We conducted experiments with 4 to 1024 mesh points, and concluded that using 256 mesh points is appropriate for architectural simulation (error was limited to 2%, also see [19]). The offline component of LightSim processes the data from Hotspot, and finds the Hankel transforms of the Green's functions for the center, rim, and corner. Subsequently, the online part computes the Hankel transform of the final temperature field for a given time interval (only for transient and transient leakage problems). It then converts the Hankel transform into a 2D FFT by converting to Cartesian coordinates. We divide the power map into three portions – center (90% of area), corners (2%), and edges (8%). We compute the 2D FFT of each power map, and convolve it with the 2D FFT of the corresponding Green's function. The final temperature field is a superposition of the three individual temperature fields. We assume 20% leakage at ambient temperature (30°C). We collect all our results on a Quad core, Intel i7 (3.1 GHz) desktop processor (memory 4GB) running Ubuntu Linux 12.1.

B. Accuracy

We compare LightSim with the popular Hotspot simulator. Figure 3 shows a plot of the Green's function for the steady state leakage problem. For Hotspot, we get the Green's function with leakage by applying a point heat source to the 136^{th} grid point (die center), and then running the simulation multiple times till convergence. For LightSim, we plot $leakspread$. Note that the x-axis shows the number of the grid point (printed row wise). The error is less than 2%. Figure 4 compares the obtained Green's functions using both the approaches for the transient leakage problem (for the origin). We observe a maximum divergence of 0.18°C (4.5%) at 1 ms. For 100 random power maps, we show the maximum error in Table 5 computed at 2ms. For Hotspot, we compute the leakage using the BSIM4 model. For a sample power map(Fig. 6), we show the results of transient simulation at 0.01 ms (Fig. 7), 0.1ms (Fig. 8), 1 ms (Fig 9), and 5 ms (Fig. 10). The y-axis shows the increase in temperature, ΔT.

C. Speed

Simulator	Steady state leakage	Transient leakage (5ms)
Hotspot 5	30.3ms	45s
CONTILTS	25.2 ms	206.4 ms
Liu et. al. [7]	39.3 ms	264.8 ms
PowerBlur	20.1ms	58.2 ms
LightSim	8.7 ms	12.8 ms

TABLE III
Comparison of execution times

We compare LightSim with Hotspot, CONTILTS, Liu et. al.[7], and PowerBlur in Table III. We simulate all the algorithms (other than Hotspot) in C. The open source C code for Hotspot 5 is freely available for download. Secondly, for the purpose of fair comparison, we precompute all the Green's functions and steady state matrices before hand. These parameters are specific to a chip, and do not change with the power map, or transient simulation interval. For the steady state case, LightSim is at least 2.3 times faster than PowerBlur. For transient simulation with leakage, LightSim is 3500 times faster than Hotspot for a 5ms interval, and is 4.5 times faster than the nearest competitor (PowerBlur). Fundamentally, LightSim is faster than Green's function based methods (PowerBlur) because it incorporates the effect of leakage, and does not require multiple iterations. Moreover, it is significantly faster than finite difference based methods because they rely on costly $O(n^2)$ or $O(n^3)$ time matrix operations, whereas LightSim uses faster FFT based methods ($O(nlog(n))$ time), and using Hankel transforms makes the computation of the Green's function a 1D problem.

V. Conclusion

We conclude that our Hankel transform based approach to quickly compute the Green's function for steady state and transient analysis with leakage, is useful for fast architecture level temperature estimation.

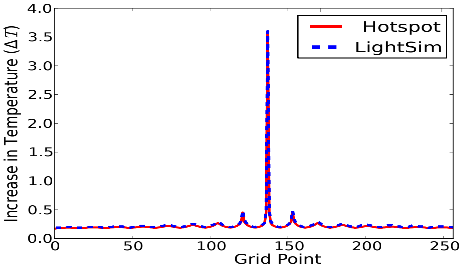

Fig. 3. Steady state Green's function

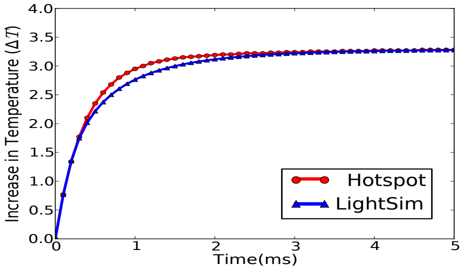

Fig. 4. Transient Green's function at the origin

Location	Error (%)	
	steady state	transient(at 2ms)
Center	0.6%	1%
Edge	1%	1.8%
Corner	1.8%	2.4%

Fig. 5. Errors of different configurations

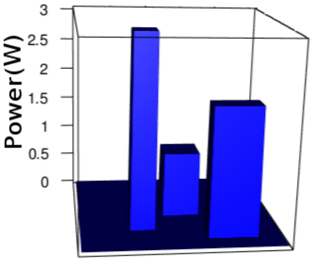

Fig. 6. Power Map

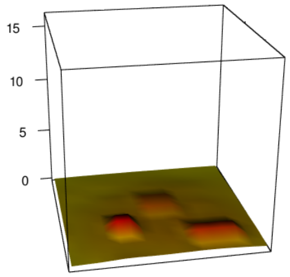

Fig. 7. ΔT at 0.01 ms

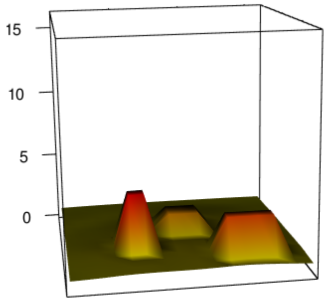

Fig. 8. ΔT at 0.1 ms

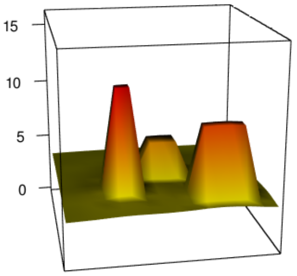

Fig. 9. ΔT at 1 ms

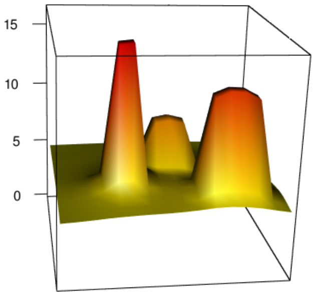

Fig. 10. ΔT at 5 ms

References

[1] S. Biswas, M. Tiwari, T. Sherwood, L. Theogarajan, and F. T. Chong. Fighting Fire with Fire: Modeling the Datacenter-Scale Effects of Targeted Superlattice Thermal Management. In *ISCA*, 2011.

[2] Elmar Gondro, Oskar Kowarik, Gerhard Knoblinger, and Peter Klein. When do we need non-quasistatic cmos rf-models? In *Custom Integrated Circuits, 2001, IEEE Conference on.*, pages 377–380. IEEE, 2001.

[3] B. Goplen and S. Sapatnekar. Efficient thermal placement of standard cells in 3d ics using a force directed approach. In *ICCAD*, 2003.

[4] Y. Han, I. Koren, and C. M. Krishna. Tilts: A fast architectural-level transient thermal simulation method. *J. Low Power Electronics*, 3(1):13–21, 2007.

[5] H. Huang, V. Chaturvedi, G. Liu, and G. Quan. Leakage aware scheduling on maximum temperature minimization for periodic hard real-time systems. *J. Low Power Electronics*, 8(4), 2012.

[6] H. Huang, G. Quan, and J. Fan. Leakage temperature dependency modeling in system level analysis. In *ISQED*, 2010.

[7] P. Liu, Z. Qi, H. Li, L. Jin, W. Wu, S. X. D. Tan, and J. Yang. Fast thermal simulation for architecture level dynamic thermal management. In *ICCAD*, pages 639–644, 2005.

[8] W. Liu, K.M. Cao, X. Jin, and Chenming Hu. Bsim 4.0.0 technical notes. Technical Report UCB/ERL M00/39, EECS Department, University of California, Berkeley, 2000.

[9] Y. Liu, R. P. Dick, L. Shang, and Huazhong Y. Accurate Temperature-dependent Integrated Circuit Leakage Power Estimation is Easy . In *DATE*, 2007.

[10] F. J. Mesa-Martinez, J. Nayfach, and J. Renau. Power Model Validation through Thermal Measurements . In *ISCA*, 2007.

[11] P. K. Mittal. *Integral Transforms for Engineers and Physicists*. Har Anand Publishers, 2007.

[12] J. Park, S. Shin, J. Christofferson, A. Shakouri, and S. Kang. Experimental validation of the power blurring method. In *SemiTherm*, 2010.

[13] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2013.

[14] K. Skadron, M. R. Stan, W. Huang, S. Velusamy, K. Sankaranarayanan, and D. Tarjan. Temperature-Aware Microarchitecture. In *ISCA*, 2003.

[15] T. Wang and C.C.P. Chen. Thermal-adi - a linear-time chip-level dynamic thermal-simulation algorithm based on alternating-direction-implicit (adi) method. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 11(4):691–700, 2003.

[16] T. Wang and C.C.P. Chen. Spice-compatible thermal simulation with lumped circuit modeling for thermal reliability analysis based on modeling order reduction. In *ISQED*, 2004.

[17] Y. Zhan and S. S. Sapatnekar. Fast computation of the temperature distribution in vlsi chips using the discrete cosine transform and table look-up. In *ASPDAC*, 2005.

[18] Y. Zhan and S.S. Sapatnekar. A high efficiency full-chip thermal simulation algorithm. In *ICCAD*, 2005.

[19] A. Ziabari, E. K. Ardestani, J. Renau, and A. Shakouri. Fast thermal simulators for architecture level integrated circuit design. In *SemiTherm*, 2011.