A Fast Compact Thermal Model For Smart Phones

Anjali Agrawal
Dept. of Electrical Engineering
IIT Delhi, India
Email: anjaliagrawal3298@gmail.com

Anand Singh
Dept. of Electrical Engineering
IIT Delhi, India
Email: anandsinghmahich@gmail.com

Ankit Gola
Dept. of Electrical Engineering
IIT Delhi, India
Email: ankitgola005@gmail.com

Hameedah Sultan
School of Information Technology
IIT Delhi, India
Email: hameedah@cse.iitd.ac.in

Smruti R. Sarangi
Computer Science and Engineering
IIT Delhi, India
Email: srsarangi@cse.iitd.ac.in

***Abstract*—In this paper, we present a fast, compact thermal model for modeling the temperature of smartphones. Existing approaches use the finite element (FEM) or finite difference (FDM) based methods that are very slow. Even fast Green's function-based approaches always use such FEM/FDM based approaches to compute the Green's function (impulse response of a power source) in the first place. This significantly slows down the process of design space exploration. To ameliorate this, we propose an ultra-fast model that can be used to model the temperature of mobile phones: we use simple polynomial or exponential expressions to compute the Green's functions. These expressions can be evaluated very quickly and can be generalized for a wide variety of electronic components. In a smartphone, analysis of the temperature hotspots is very crucial in the design process. We can estimate the location of hotspots and the temperature rise at those hotspots with very high accuracy. Our error is limited to 2.56% and our tool is 1300 times faster than the nearest competing, state-of-the-art tool.**

I. INTRODUCTION

Significant improvements in device performance and functionality combined with affordable costs have resulted in a huge demand for smartphones and tablets in recent years. The processor speed has increased manifold, but the voltage requirements of the device haven't scaled down accordingly. This has resulted in increased power dissipation in the device [1]. Higher power dissipation causes many adverse effects such as higher temperatures in the die, which degrades performance and reliability [2]. Furthermore, as the PCB temperature increases, so does the temperature on the screen of the device (also called the *skin temperature)*. This is a serious problem since users of hand-held mobile devices can tolerate heat only up to a certain limit, beyond which it can degrade the user experience [3] and can even prove to be dangerous in terms of battery explosions.

High skin temperatures beyond 45°C can result in a severely degraded user experience [4]: it is a limit that is easily reached in small form-factor mobile devices. Consequently, it is important to keep the skin temperature under limits (about 41°C for an aluminum casing and 45°C for a plastic casing [5]). At present, this is achieved by throttling the processors: reducing performance to keep the temperature under limits. However, this results in a sub-optimal design. The optimal approach requires a coordinated *system-level* thermal design and management solution that encompasses the hardware as well as the software [6]. The structure of the system, its physical layout, material properties, and power dissipation all affect the thermal profile. However, to assess the impact of these design decisions and find an optimal solution, an ultra-fast *system-level* thermal simulation method is needed. Unfortunately, the research in this area is very sparse and inadequate. The ability to devise an optimal thermal design that maximizes performance is contingent on fast and accurate thermal simulations, and throughout the design cycle, a very large number of such simulations need to be done. Hence, the speed of a thermal simulator is extremely important and even a small speed-up results in huge savings in time and effort.

While there is a large body of work focused on thermal simulation at the level of processors, memories, and MPSoCs, a very limited number of techniques exist at the system-level[1]. Moreover, these techniques use the traditional finite element (FEM) or finite difference (FDM) methods to solve the classical Fourier heat equation and compute the temperature profile. However, these methods are quite slow, especially for complete system-level thermal analysis. Moreover, they do not capture the complex relations between various system components accurately [7]. A much faster category of thermal simulators is based on Green's functions [8]–[10]. Here, the impulse response (Green's function) is first obtained, and then at the runtime, this impulse response is convolved with the power map to compute the complete temperature map. However, such an approach cannot be trivially extended to the system-level, because of the complex relationship between the various system components [7].

The main issue with Green's functions is that we need to compute them using a slow method: physical measurements or FEM/FDM simulations. It is often the case that we need to simulate the thermal profile for many different physical layouts and material properties. In such cases, we need to recompute the Green's functions for every single configuration using rather slow methods. This nullifies the performance gains of using Green's functions. Consequently, in this work, we develop a very fast method to estimate the Green's functions themselves. This is a novel approach and to the best of our knowledge, it has not been tried before. We propose a model based on small and simple functions – ultra-compact thermal models in our parlance – to help us calculate the Green's function for a given layout and configuration. These Green's functions can subsequently be used to simulate the thermal profile: this part is very fast.

1) The first step was to create a simulation setup that is calibrated with real-world data. We performed three-way cross-validation of infra-red (IR) images of devices, thermocouple based measurements, and CFD models (using Ansys Icepak).

2) We then developed a system-level Green's function-based thermal modeling approach by superposing the component level temperature maps obtained using classical Green's function approaches.

3) To reduce the time needed to obtain the Green's function, we developed an ultra-compact thermal model that relies on simple

[1] By system, we mean complete electronic systems with a small form factor, such as tablets and smartphones.

polynomial expressions to very quickly compute the Green's function for a wide range of design choices.

We demonstrate that the error using our algorithm is limited to 2.56% while simultaneously being 1300 times faster than the state of the art approaches. Moreover, our ultra-fast thermal modeling method makes it possible to sift through a very large number of configurations to obtain the optimal configuration quickly – something which would be prohibitively expensive with the traditional approaches used by the state of the art tools.

To the best of our knowledge, we are the first to develop a three-way calibrated CFD model. We are also the first to develop a Green's function-based system-level thermal modeling method, as well as an ultra-compact thermal model that doesn't need to compute these Green's functions are runtime.

In section II, we introduce a relevant background. We describe our IR imaging framework and the creation of CFD models calibrated with IR images in Section III. Next, we describe our Green's function-based modeling methodology in Section IV. We have evaluated our algorithm and presented the results in Section V. Section VI concludes this paper.

II. Background

A. Heat Transfer

Heat transfer in solids is mainly governed by the Fourier's heat equation given by

$$\nabla . [\kappa \nabla T] + \dot{q} = \rho C_v \frac{\partial T}{\partial t}, \tag{1}$$

where T is the temperature, C_v is the volumetric specific heat, t is the time, q is the heat flux, and κ is the thermal conductivity of the material. Most thermal simulation methods either use the finite element method (FEM) or the finite difference method (FDM) to solve this differential equation.

B. Green's Functions

An alternative and faster approach for computing the thermal profile is based on Green's functions [8]–[10]. A Green's function is the impulse response of a unit power source (Dirac delta function) applied to the center of the chip. The temperature distribution for any power distribution can be quickly found using a convolution of the Green's function with the input power map of the system. This stems from the principles of linearity and shift-invariance (the Green's function is the same across a neighborhood).

$$T = P \star G, \tag{2}$$

where T is the temperature profile, P is the power map of the system, and $\star$ is the 2D convolution operator.

C. Related work and Ultra-Compact Thermal Models

Several researchers have attempted to analyze the effects of major device components on skin temperature. Xie et al. [11] show that a significant thermal coupling effect exists between major heat-generating components, such as battery and application processor (AP) and they propose a dynamic thermal management (DTM) method considering this effect. Therminator is the most related work [12], [13]; it uses the finite difference method to calculate the thermal profile of the device. We shall show that it is 1300 times slower than our approach. Sadiqbatcha et al. [14] propose a method to identify the major heat sources in IR images of a multicore processor and propose a learning-based dynamic thermal model to predict the temperature of identified heat-generating sources at runtime. Park et. al [7] propose a method for skin temperature estimation by using different models for different types of components and considering the interaction between components such as the LCD screen and the battery. The temperature models are simply based on the cubic power-frequency relations. The authors in [15] have created detailed FEM models in Ansys for mobile devices and studied the thermal effects of different material combinations with different conductivities. Satomi et. al [16] propose a genetic algorithm for thermal-aware placement of components on the PCB. However, their method for temperature estimation is very simplistic and has only one 3D thermal resistance for an entire component. Egilmez et. al [3] collect 13 features such as CPU utilization and temperature from thermistors and then apply machine learning algorithms to predict the temperature. They then use this prediction model to keep the skin temperature under check using DVFS. However, learning-based algorithms are very sensitive to the nature of the data, and often don't generalize well.

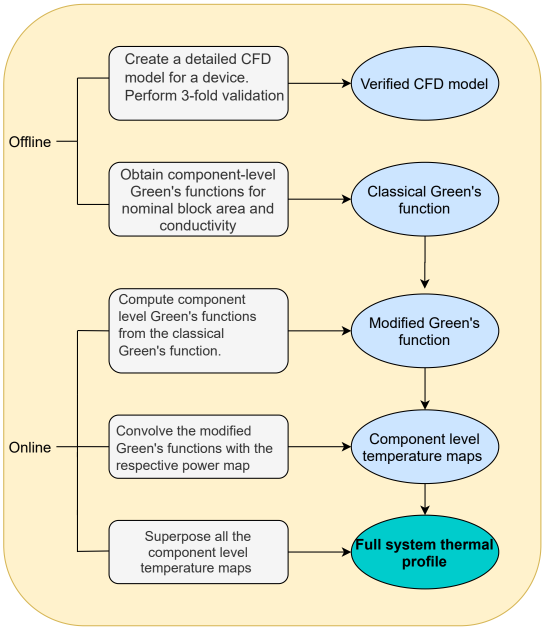

Offline
Create a detailed CFD model for a device. Perform 3-fold validation
Verified CFD model
Obtain component-level Green's functions for nominal block area and conductivity
Classical Green's function
Online
Compute component level Green's functions from the classical Green's function.
Modified Green's function
Convolve the modified Green's functions with the respective power map
Component level temperature maps
Superpose all the component level temperature maps
Full system thermal profile

Fig. 1: Overview of our approach

The main drawback of all of these methods is that they make very coarse approximations and are additionally much slower than our solution. Our tool can rapidly explore thousands of configurations (layout+material properties) in less than 10 seconds.

III. IR Imaging Framework

A. Overview of our Approach

We employ a novel two-step abstraction process to develop an ultra-fast compact thermal model of a complete electronic system. We describe an overview of our method first:

1) a) In the first step, we create a novel non-intrusive IR imaging framework.
   b) Next, we create an accurate CFD model for the device by calibrating it against the real IR images and thermocouple readings. This three-way cross-validation enables accurate parametric modeling for a wide range of properties.
2) a) Subsequently, we create a Green's function-based thermal model for the complete system.

b) We then create an ultra-compact thermal model based on simple polynomial expressions to compute the Green's function for different components very quickly.

A flowchart describing an overview of our method is shown in Figure 1. We describe the IR imaging framework and the equivalent CFD models in this section. We describe the Green's function-based ultra-compact thermal modeling method in Section IV.

B. CFD Models and Overview of the IR Imaging Framework

CFD models are necessary for two reasons:

1) The device specifications such as component dimensions, placement, and material properties are varied at design time to obtain an optimal configuration. It would not be practically possible to vary such properties in real devices and hence an equivalent CFD model is necessary.
2) To empirically obtain the Green's function, a unit impulse source has to be applied to the center of the chip. However, a unit power source in an individual component cannot be practically applied in a running system. The best bet in such a case is a CFD model verified against real IR images, which offers a lot of design flexibility.

We use the industry-standard tool Ansys Icepak for CFD modeling. It uses the Fluent solver for thermal and fluid flow analyses and applies the finite element method to solve the heat flow equations. We create three separate CFD models for a smartphone, tablet, and laptop, which have been validated against the IR images of the respective devices. To do so, we capture IR images of the target device under different stress conditions (resulting in different power dissipation profiles). We extract the per-component power values corresponding to these stress conditions using a novel non-intrusive approach. These IR images need to be pre-processed to extract the thermal maps. Finally, we validate the CFD models against these known power and thermal maps. A schematic diagram of the Icepak model for one of the devices is shown in Figure 3.

1) Workloads: There are multiple power dissipating components in any electronic system, such as the processor, RAM, flash memory, hard disk drive (HDD). However, most of these components can be classified into two categories: *compute-intensive,* and *memory-intensive.* To generate the thermal maps corresponding to different workloads applied to these components, we have designed some workloads in Python, which we call stress scripts, that exercise the individual components. The power consumption corresponding to that element is then calculated.

Stress Test of Processing Components: This stress test is designed to stress the computing element by maximizing its power consumption. The user is required to enter the average number of threads it wishes to spawn per core – the *thread multiplier*. Then it starts a process that spawns as many threads as the number of cores multiplied by the thread multiplier entered by the user. Each thread then starts counting from three and tests whether the current number is prime or not. We do not implement any optimization in this process as we aim to stress the cores maximally.

Stress Test of Memory Components: To stress the memory, the script asks the user to supply the maximum amount of disk space it wants the test to take and the number of iterations. The test then starts a process that copies data from the special file *uzero* to the flash memory. After one iteration of writing, the script runs a cleanup job where it wipes the caches and deletes the file it just wrote before starting the next iteration. Once all the iterations are over, it returns the time taken for running the test in microseconds. To address the issue of the lack of direct access to the RAM in Android, we map a disk folder directly over the RAM using the tool *RAMdisk*. We limit the maximum write size to 1 GB.

The total power consumption for running the tests is found either through a power meter for laptops or using the power profiling tool TREPN [17] from Qualcomm for Android devices. When we execute the stress tests for memory elements, the processor too consumes some power. To determine the power consumed solely by the memory component, we first accurately measure the power consumed by the processor by running a program that does not have the memory instructions added to stress the system, and then we subtract this power from the results of the memory stress test.

2) Pre-processing of the IR Images: We use image processing to extract the region of interest (ROI) in the image corresponding to the device in an automated manner. For this, we have used the ROI as a mask. This masking technique sets all other pixel values except the ones inside the ROI to zero. To find the ROI, we used the Canny edge detector. First, we applied a Gaussian blur to smoothen the image. To detect the edge intensity, we calculated the gradient of the image by filtering using a Sobel kernel. To thin out the edges, we used non-maximum suppression.

Next, we needed to calibrate the IR camera itself. This is because IR cameras detect radiations emitted from the target body. This detected radiation depends on several factors such as the emissivity of the target object and lens configurations of the camera. For calibration, we used a pre-calibrated thermocouple to accurately measure the ambient temperature and obtain IR images at multiple ambient temperatures. We then used the following equation to calibrate our IR camera:

$$T_{Therm} = C_a T_{IR} + C_b \tag{3}$$

where, T_{Therm} is a known temperature reading obtained using the thermocouple, T_{IR} is the IR camera reading, and C_a and C_b are constants. We obtained measurements for multiple ambient temperatures, and then fit a curve to obtain the values of the constants C_a and C_b. Figure 2 shows our infrared imaging setup.

IV. Green's Function-Based Ultra-Compact Thermal Modeling

A. System-Level Thermal Modeling using Green's Functions

At the system-level, multiple components are present on a PCB and they have different Green's functions. Hence, for each power dissipating component, we obtain the Green's function for that component using Ansys Icepak by applying a unit power source to the center of the individual component. Since the complete system is linear, the principle of superposition holds. So we convolve the component-level Green's function with the component-level power profile and

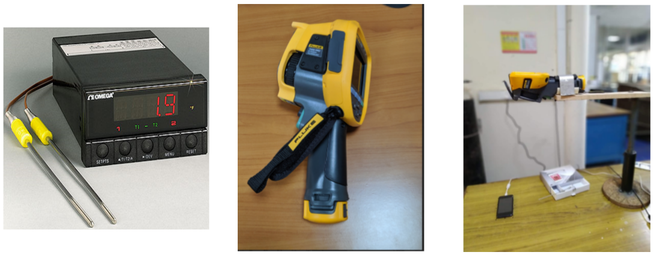


Fig. 2: Devices used and the setup for calibration (a) Digital Temperature Sensor (b) Thermal Imaging Camera (c) Experimental setup

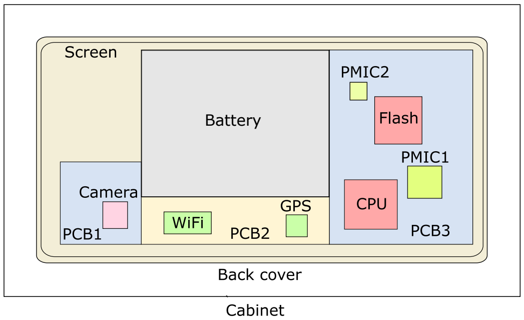


Fig. 3: Layout of one of the test devices

TABLE I: Glossary

Symbol	**Meaning**
A	Area
κ	Conductivity
a, b, c, d, e	Coefficients of the fitted model
G_{base}	Baseline Green's function, obtained at nominal area and conductivity
C_{comp}	Multiplier to Baseline Green's function
$G_{mod}(A, \kappa)$	Modified Green's function
P	Power map of individual component
T_{final}	Complete system's temperature profile

superimpose the resultant temperature profiles for all the components. This gives us an accurate complete system thermal profile. We call the Green's functions so obtained as the baseline Green's functions, G_{base}, and the corresponding area and conductivity as the nominal area and conductivity. Table I lists the symbols used in our work. The problem is to compute the modified Green's functions G_{mod} for different areas and conductivities. We approximate $G_{mod} = C_{comp} \times G_{base}$ (justified empirically in Section V). We thus need to estimate C_{comp} for a given area and thermal conductivity.

B. Compact Thermal Modeling using Green's Functions

We first collect a large amount of temperature data by varying a block's (CPU, flash, etc.) parameters. We see that the parameters that have the most significant effects on temperature are the area (A) and thermal conductivity (κ). Hence, we try to find C_{comp} as a function of A and κ.

1) Parametric Model for the Power Dissipating Components (e.g. Processor, Flash/HDD, RAM): We consider several polynomial and exponential expressions to fit C_{comp}. Table II shows a set of candidate functions along with their root mean squared error (RMSE) and the R-squared (R^2) values for two representative components: processor and flash. Similar results were obtained for the other components too. R^2 represents the ratio of the variation in the data explained by the fit model to the total variation in the data. It lies between 0 and 1. The higher its value, the better is the fit. We see that C_{comp} is quadratically dependent on the area and linearly dependent on the conductivity. This is because the temperature profile of a block has a higher dependency on the variation of the area as compared to the variation of conductivity.

Thus, we obtain a second-order polynomial expression for C_{comp} that is multiplied with the baseline Green's function G_{base}.

$$C_{comp} = a + bA + c\kappa + dA^2 + eA\kappa \tag{4}$$

$$G_{mod} = C_{comp} \times G_{base} \tag{5}$$

TABLE II: Comparison of different compact models for the processor and flash memory

Parametric Model	Processor		Flash	
	R^2	RMSE (°C)	R^2	RMSE (°C)
$ae^{bA} + ce^{d\kappa}$	0.79	0.06	0.90	0.05
$a^{A/b} + c^{\kappa/d}$	0.87	0.05	0.93	0.04
$(aA + b)/(cA + d)$	0.91	0.04	0.93	0.04
$a + bA + cy$	0.91	0.04	0.93	0.04
$a + b\ln(A/\kappa)$	0.95	0.03	0.96	0.03
$a+bA+c\kappa+dA^2+eA\kappa$	0.99	0.02	0.99	0.01

TABLE III: Comparison of Different Models taken for the non-silicon components

Model	R^2	RMSE (°C)
Exponential	0.98	7.05 $\times e^{-5}$
Polynomial	0.97	8.70 $\times e^{-5}$
Power	0.98	8.07 $\times e^{-5}$
Rational	0.97	1.03 $\times e^{-4}$

where A is the area of the component, κ is the conductivity, G_{mod} is the baseline Green's function obtained at the nominal conductivity and area, G_{base} is the modified Green's function and C_{comp} is the multiplier to the baseline Green's function.

2) Parametric Models for the Other Components: For the non-silicon components such as the battery, we observe that the conductivity doesn't have as much of an effect on the temperature as the area. Hence, to minimize the model complexity, we only consider one variable: the area of the component. Table III lists the RMSE for the different models explored. We choose the model with the least RMSE:

$$C_{comp} = ae^{bA} \tag{6}$$

We next compute the temperature profile of each component by convolving the modified Green's function with the respective power map. To compute the temperature profile for the complete system (T_{final}), we superimpose the individual temperature maps as per Equation 7.

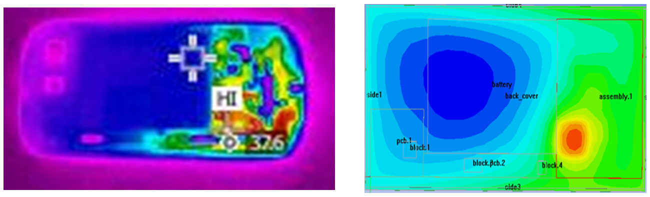
$$T_{final} = \sum_{n=1}^{n} G_{mod,i} \star P_i \tag{7}$$

where n is the number of components in the system, $G_{mod,i}$ is the modified Green's function for the i^{th} component, P_i is the power map of the i^{th} component, and $\star$ is the 2D convolution operator.

V. Evaluation

A. Setup

1) IR Imaging Setup: We use the Fluke Ti450 pro thermal imaging camera to obtain the IR images. It is an industrial-grade thermal camera with a sensor resolution of 320 × 240 for a total of 76800 pixels. To calibrate the IR camera we perform multiple measurements at different ambient temperatures and fit the results into Equation 3. The best fit curve gives $C_a = 1.17$ and $C_b = -5.93$. We obtain the IR images for three devices: Samsung Galaxy S3, Lava IRIS X1 mini, and Lenovo Thinkpad.



(a) IR camera thermal map

(b) Icepak thermal map

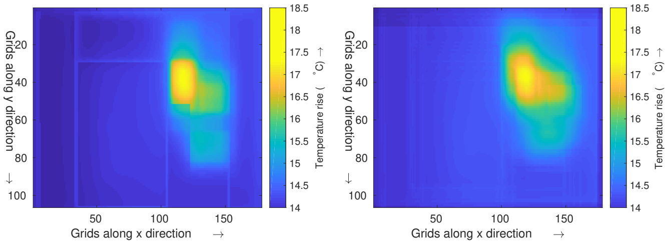
Fig. 4: Comparison of Icepak and IR outputs for the S3 phone

2) Green's function setup: We run the Icepak simulations on a desktop running Ubuntu Linux 16.04 with the Intel $i5-4950$ processor working at 3.30 GHz and 12 GB RAM. For the CFD simulations, we used Ansys Icepak (version 19.2). The scripts for our approach were written in MATLAB R2017b.

B. Verification of the Icepak Models

We design an Icepak model for each device. Next, we obtain the IR images for different stress tests and measure the corresponding power dissipation using the approach described in Section III-B1. We then apply the same power to our Icepak models and compare the temperature maps against the corresponding pre-processed IR images. The pre-processing of IR images is done in Python. We observe an average error of 5% meaning that our Icepak models represent the actual devices fairly accurately. Figure 4 shows the thermal profiles obtained from Icepak as compared to the IR images for the Samsung Galaxy S3. The average error, in this case, is 1.4%.

C. Green's Function-based Thermal Modeling



(a) Icepak thermal profile

(b) Calculated thermal profile

Fig. 5: Full-System temperature map using classical Green's functions

1) Classical Green's-function Approach with Superposition: In Figure 5, we show the actual thermal map obtained using Ansys Icepak in the presence of multiple power sources alongside the calculated thermal map using the standard Green's function approach. We store one Green's function per component and perform its convolution with the respective power map to get the component level thermal profile. Superimposing all such thermal profiles gives the full system temperature map. This takes approximately 40.8 ms and yields the maximum screen temperature with an error of less than 0.5%. This shows that the Green's function-based approach works well at the system level and can quickly and accurately give us the thermal profile. The error reported in this work is the error in computing the maximum system temperature on the screen of the device.

2) Compact thermal modeling: First, we collect thermal data by taking 45 different area and conductivity values for each component. We vary the area of the processor between 49 mm^2 and 225 mm^2. The thermal conductivity was varied from 120 W/mK to 150 W/mK. The area of the flash memory was varied from 36 mm^2 to 169 mm^2 and the thermal conductivity was varied from 120 W/mK to 150 W/mK. We considered the nominal CPU area to be 144 mm^2 and the nominal flash area to be 156 mm^2. The area of the battery was varied from 2656 mm^2 to 3189 mm^2. Note that the thermal conductivity, in this case, captures the conductivity of the package itself: even though the silicon die's conductivity varies very little, the packaging material keeps changing.

TABLE IV: Constants for three representative components

Component	a	b	c	d	e
CPU	1.49	$-5.07e-3$	$3.63e-5$	$1.13e-5$	$-2.29e-7$
Flash	1.63	$-7.46e-3$	$-5.4e-5$	$2.13e-5$	$-3.77e-8$
Battery	1.007	$-2.37e^{-6}$	–	–	–

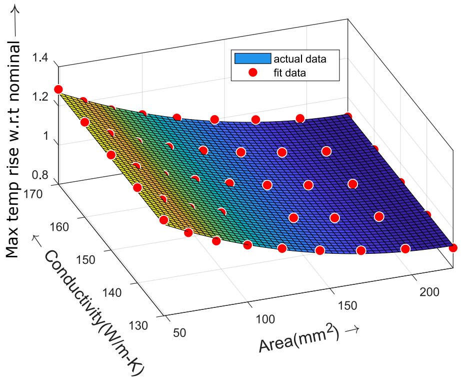
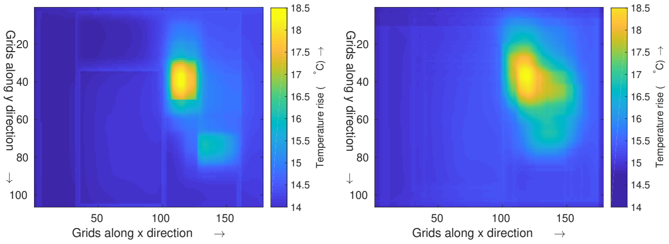


Fig. 6: Maximum temperature rise of the CPU relative to nominal CPU properties as a function of area and conductivity

Next, we fit a second-order polynomial model for the measured data. Figure 6 shows the variation of the maximum CPU temperature relative to that of the nominal CPU as a function of the area and conductivity. The root mean square error is 0.017°C for the CPU and 0.015°C for flash. The maximum temperature rise for a nominal CPU is 12.75°C and the error in computing the modified Green's function for a CPU is 0.8%. The area and conductivity limits were chosen in accordance with values considered in the literature. The values of C_{comp} for the CPU, flash, and battery are shown in Table IV.



(a) Icepak thermal profile

(b) Calculated thermal profile

Fig. 7: Full-System Temperature Map using Modified Green's functions

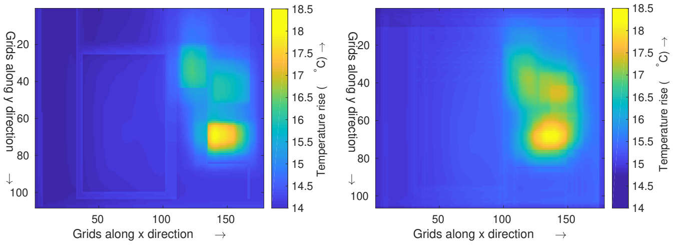
Complete System-level Thermal Profile: Figure 7 shows the comparison of the Icepak thermal map with the calculated thermal map when the component areas and conductivity values are varied at runtime. We report an error of 2.56% for the maximum temperature

TABLE V: Comparison with other system-level simulators

Approach	Accuracy	Speed
Therminator [13]	98%	55 s
Our approach	97.4%	0.041 s

rise of the system. Our method takes only 41 ms to compute the full-system thermal profile. In addition to this, we accurately predict the location of all the hotspots in the system.

3) Generalizability to Multiple Devices: We next implement our modeling method for new device layouts to see how well our method generalizes. Figure 8 shows the comparison between the thermal profile obtained using Ansys Icepak and the calculated thermal profile for one such layout in which the positions of a few components are changed. Our approach computes the thermal profile with an error of 1.33%. This demonstrates that our method can be used to model a variety of device layouts without having to recompute the Green's function using slow FDM or FEM simulations. This is very useful in the design phase, where the thermal characteristics of several candidate layouts can be quickly studied.



(a) Icepak thermal profile (b) Calculated thermal profile

Fig. 8: Full-system temperature map using modified Green's functions

D. Comparison with state-of-the-art simulators

CFD simulators such as Ansys Icepak are capable of providing a detailed system-level thermal profile. However such methods are very slow, sensitive to the meshing, and prone to convergence issues. In our models, Icepak approximately requires 8 minutes to compute the full system thermal profile. The total time that our model takes to compute the complete thermal map of the system comes out to be less than $41ms$. This includes the time taken for scaling the Green's functions, convolving the modified Green's functions with respective power maps, and the time taken to superimpose all the individual thermal maps. Thus our algorithm provides over a $12,000$ times speedup over Icepak. We compare our results with the closest accurate system-level thermal simulation tool, Therminator [12], [13]. It is based on the finite difference method and extends the methodology of the popular thermal modeling tool Hotspot [18] to the system-level. It needs 55 s to compute the thermal map across 5313 points with the same number of power dissipating components as our system. In comparison, we consider more than 18,000 meshing points and take only 41 ms to compute the full thermal map (a speedup of 1300 times). Moreover, our approach has been validated by thermal measurements on real hardware. In Table V, we present a comparison of our approach with Therminator for the speed and accuracy (maximum system temperature on the screen). The accuracy is almost the same.

VI. CONCLUSION

In this paper, we proposed an ultra-compact model for system-level thermal simulations of smartphones. Our method is capable of modeling the detailed thermal profile very quickly because of the novel approach used. First, we validate our model with an IR Camera and thermocouple based measurements by comparing the temperature maps and report an error of less than 5%. Next, we develop a compact thermal model that can predict the location of the hotspots and compute the maximum temperature rise at those hotspots with an error of 2.56% . Our method is 1300 $times$ faster than the state of the art thermal simulators and has been validated with a three-way cross-validation methodology. Such approximations using simple polynomial and exponential functions for the Green's functions had not been done in the past. It was believed that such simple approaches will not be effective. However, we show that such approaches can indeed be very accurate and effective.

REFERENCES

[1] J. M. Rabaey, *Digital Integrated Circuits: A Design Perspective*. Prentice-Hall, Inc., 1996.

[2] D. Lackey, P. Zuchowski, T. Bednar, D. Stout, S. Gould, and J. Cohn, "Managing power and performance for system-on-chip designs using voltage islands," in *ICCAD*, 2002.

[3] B. Egilmez, G. Memik, S. Ogrenci-Memik, and O. Ergin, "User-specific skin temperature-aware dvfs for smartphones," in *DATE*, 2015.

[4] B. Averbeck, L. Seitz, F. P. Kolb, and D. F. Kutz, "Sex differences in thermal detection and thermal pain threshold and the thermal grill illusion: a psychophysical study in young volunteers," *Biology of sex differences*, vol. 8, no. 1, p. 29, 2017.

[5] M. K. Berhe, "Ergonomic temperature limits for handheld electronic devices," in *International Electronic Packaging Technical Conference and Exhibition*, 2007.

[6] V. Chiriac, S. Molloy, J. Anderson, and K. Goodson, "A figure of merit for smart phone thermal management," *Electronics Cooling*, vol. 17, pp. 18–23, 2015.

[7] J. Park, S. Lee, and H. Cha, "Accurate prediction of smartphones' skin temperature by considering exothermic components," in *DATE*, Apr. 2018.

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

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

[10] H. Sultan, A. Chauhan, and S. R. Sarangi, "A survey of chip-level thermal simulators," *CSUR*, vol. 52, no. 2, pp. 1–35, 2019.

[11] Q. Xie, J. Kim, Y. Wang, D. Shin, N. Chang, and M. Pedram, "Dynamic thermal management in mobile devices considering the thermal coupling between battery and application processor," in *ICCAD*, 2013.

[12] Q. Xie, M. J. Dousti, and M. Pedram, "Therminator: a thermal simulator for smartphones producing accurate chip and skin temperature maps," in *ISLPED*, 2014.

[13] M. J. Dousti, M. Ghasemi-Gol, M. Nazemi, and M. Pedram, "Thermtap: An online power analyzer and thermal simulator for android devices," in *ISLPED*, 2015.

[14] S. Sadiqbatcha, H. Zhao, H. Amrouch, J. Henkel, and S. X. . Tan, "Hot spot identification and system parameterized thermal modeling for multi-core processors through infrared thermal imaging," in *DATE*, 2019.

[15] J. Lee, D. Gerlach, and Y. Joshi, "Parametric thermal modeling of heat transfer in handheld electronic devices," in *ITHERM*, 2008.

[16] Y. Satomi, K. Hachiya, T. Kanamoto, R. Watanabe, and A. Kurokawa, "Thermal placement on pcb of components including 3d ICs," *IEICE Electronics Express*, 2020.

[17] T. Profiler, "Trepn power profiler," 2017.

[18] W. Huang, S. Ghosh, S. Velusamy, K. Sankaranarayanan, K. Skadron, and M. Stan, "Hotspot: a compact thermal modeling methodology for early-stage vlsi design," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 14, pp. 501–513, 2006.