Therminator: A Thermal Simulator for Smartphones Producing Accurate Chip and Skin Temperature Maps

Qing Xie, Mohammad Javad Dousti, and Massoud Pedram
University of Southern California, Los Angeles, CA, USA
{xqing, dousti, pedram}@usc.edu

ABSTRACT

Maintaining safe chip and device skin temperatures in small form-factor mobile devices (such as smartphones and tablets) while continuing to add new functionalities and provide higher performance has emerged as a key challenge. This paper presents *Therminator*, an early stage, fast, full-device thermal analyzer, which generates accurate steady-state temperature maps of the entire smartphone starting from the Application Processor and other key device components, extending to the skin of the device itself. The thermal analysis is sensitive to detailed device specifications (including its material composition and 3-D layout) as well as different use cases (each case specifying the set of active device components and their activity levels). Therminator considers all major components within the device, builds a corresponding compact thermal model for each component and the whole device, and produces their steady-state temperature maps. Temperature results obtained by using Therminator have been validated against a commercial computational fluid dynamics-based tool, i.e., Autodesk Simulation CFD, and thermocouple measurements on a Qualcomm Mobile Developer Platform. A case study on a Samsung Galaxy S4 using Therminator is provided to relate the device performance to the skin temperature and investigate the thermal path design.

Categories and Subject Descriptors

C. 5. 3 [**Computer System Implementation**]: Microcomputers – *Portable devices (e.g., laptops, personal digital assistants)*

Keywords

Smartphones, embedded systems, thermal management, thermal modeling, temperature maps simulator, skin temperature, CFD

1. INTRODUCTION

The popularity of mobile devices, such as smartphones and tablets, has surpassed that of personal computers, thanks to their portability and ease-of-use. (In the remainder of this paper we will use smartphones as the popular and archetypical mobile device.) Additional enablers for the rapid increase in the number of smartphones have been their improving functionality and ever-increasing performance capabilities. This has in turn happened due to introduction of high performance (heterogeneous, multi-core) processors inside smartphones. Unfortunately, high performance processors cause two adverse effects: 1) They tend to experience higher average and peak die temperatures. 2) They tend to result in higher device *skin (surface) temperatures*. High die temperature increases the leakage power consumption [1], speeds up aging processes [2], and may eventually cause permanent defects. High skin temperatures can cause first or even second degree burns on device users, with obvious and immediate adverse user reactions. Hence, thermal design (i.e., designing the heat flow path and a cooling method) and thermal management (i.e., employing thermal response mechanisms to avoid hot spots and high die temperatures) are crucial for a mobile device to improve its performance and energy efficiency while maintaining safe temperatures.

Proper thermal design effectively removes heat away from a VLSI circuit die. In smartphones, *application processors* (APs) incorporate CPU, GPU, *digital signal processor* (DSP), sometimes a baseband radio unit, and so on. The AP is a major heat generator in the smartphone [3]. Due to the cost, form factor, noise, and safety issues, smartphones rely on passive cooling methods that dissipate the heat generated by the AP through thermal conduction to the device skin. Thermal pads are usually attached on top of the AP chip package to ease the heat removal [3][4]. Thermal management techniques, such as frequency throttling and voltage/frequency scaling, are also exploited to avoid high die temperatures. For example, one can observe that the CPU and GPU performance (and consequently their power consumption) are throttled in Samsung Exynos 5250 so as to prevent the AP's junction temperature from exceeding an upper threshold [5].

As noted above, thermal design and management of smartphones are also concerned a *skin temperature constraint*. This constraint refers to the fact that the temperature at the device skin must not exceed a certain upper threshold. According to [6][7], most people experience a sensation of heat pain when they touch an object hotter than 45˚C. Ideally speaking, distributing the heat uniformly onto the device skin results in the most effective heat dissipation. However, in practice, majority of the heat flows in vertical direction from the AP die, and thus hot spots are formed on the device skin above the AP location [8]. It is reported that the hottest spot on iPad 3 can reach as high as 47˚C while playing graphic intensive games [9]. Usually, a skin temperature thermal governor is implemented to maintain the skin temperature at a desired *setpoint* by using a control feedback.

To address this design challenge, it is necessary to model the temperature map (temperature at different locations) for the smartphone in an accurate and efficient manner. Knowing the detailed temperature map on the device skin at the design time is helpful in the device implementation. For example, using materials with high thermal conductivity in the thermal path enhances heat removal from the AP and in turn causes high skin temperature, whereas using low thermal conductivity materials cannot remove the heat from the AP fast enough and hence the die temperature goes up. Moreover, knowing how the temperature of a particular component depends on use cases helps to derive the optimal thermal management policy for that component. For instance, setting CPU frequency throttling levels is affected by how skin temperature depends on the CPU frequency.

Analyzing temperature maps at the early stage of the design flow can significantly reduce the device time. Even though *computational fluid dynamics* (CFD) tools generate accurate temperature maps, they are expensive and not compatible with other performance/power simulators. *Compact thermal modeling* (CTM) method has been proposed for thermal analysis with reasonable accuracy and low computational complexity [10][11]. This method

ISLPED'14, August 11–13, 2014, La Jolla, CA, USA.

http://dx.doi.org/10.1145/2627369.2627641

builds an RC thermal network based on the well-known duality between the thermal and the electrical phenomena, and solves for temperatures in the network in a similar way to finding voltage values in an electrical circuit.

In this work, we present *Therminator*, a CTM-based component-level thermal simulator targeting small form-factor mobile devices (such as smartphones and tablets). Major contributions of this work are the following:

1) Therminator is the first thermal simulator targeting at smartphones. It produces temperature maps for all components, including the AP, battery, display, and other key device components, as well as the skin of the device, with high accuracy and fast runtime. Therminator results have been validated against thermocouple measurements on a Qualcomm Mobile Developer Platform (MDP) [12] and simulation results generated by Autodesk Simulation CFD [13].
2) Therminator is very versatile in handling different device specifications and component usage information, which allows a user to explore impacts of different thermal designs and thermal management policies. New devices can be simply described through an input specification file (in XML format).
3) Therminator supports parallel processing, allowing users to employ GPU to reduce the runtime by more than two orders of magnitude for high-resolution temperature maps.
4) A detailed case study has been conducted for Samsung Galaxy S4 by using Therminator. The temperature results relate the device performance to the device skin temperature, as well as the impact of the thermal path design.

Therminator is available for download at http://atrak.usc.edu/downloads.

The rest of paper is organized as follows. Section 2 reviews related work. Section 3 introduces Therminator. The modeling methodology and implemented features are elaborated in Sections 4 and 5. We validate Therminator results in Section 6 and provide a case study in Section 7. Section 8 concludes the paper.

2. RELATED WORK

HotSpot [11] is a successful early-stage CTM methodology targeting thermal analysis of the silicon die and its packaging which are cooled with a heat sink and possibly a fan. It generates accurate temperature maps quickly. *Temptor* [14] is a tool based on HotSpot which allows the temperature prediction using performance counters instead of components' power trace. Meng *et al.* [15] improved HotSpot by adding the 3-D chip simulation support. *Teculator* [16] instruments HotSpot to support thermoelectric coolers. *3D-ICE* [17] is another thermal simulator targeting 3-D ICs equipped with liquid cooling. However, neither HotSpot nor 3D-ICE can be modified or extended to analyze small form-factor devices as they target a single IC package along with its cooling equipment. In fact, modeling smartphone is much more complicated due to: 1) multiple heat generators, including battery, display, and a number of IC chips; 2) complex 3-D layout where each component may be in vertical and horizontal contact with several other components; and 3) necessity of considering the internal air in the device. Comparing to those tools, Therminator focuses on component-level thermal modeling, in which the architecture-level details inside a single chip package are ignored.

Several researches have been conducted in studying the thermal design for smartphones and tablets [3][4][18]. Luo *et al.* established a simple thermal resistance network to analyze the whole mobile phone system [18]. However, the thermal resistance network built in [18] is oversimplified as each component is modeled as one block with a single uniform temperature value. Gurrum *et al.* modeled the smartphone in CFD tools and analyzed the thermal effect of using materials with different thermal conductivities through CFD simulation [3]. Rajmond and Fodor [4] used CFD tools to show that attaching thermal pad on top of the AP significantly reduces the AP temperature. To the best of our knowledge, Therminator is the first tool targeting smartphones that automatically builds a compact thermal model from the device specifications, and solves for temperature maps of all components accurately with a fast runtime.

3. THERMINATOR OVERVIEW

Figure 1 depicts the overview of Therminator. Therminator takes two input files provided by users. The `specs.xml` file describes the smartphone design, including components of interest and their geometric dimensions (length, width, and thickness) and relative positions. Therminator has a built-in library storing properties of common materials (i.e., thermal conductivity, density, and specific heat) that are used to manufacture smartphones. In addition, users can override these properties or specify new materials through the `specs.xml` file. The `power.trace` file provides the usage information (power consumption) of those components that consume power and generate heat, e.g., ICs, battery, and display. The `power.trace` can be obtained through real measurements or other power estimation tools/methods such as [19][20]. `power.trace` is a separate file so that one can easily interface a performance-power simulator with Therminator.

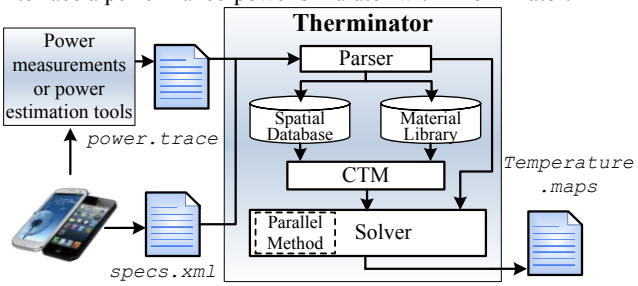

Figure 1. Overview of Therminator.

Therminator has three main modules. A *parser module* parses input files, updates the *material library*, and makes a set of components specified by the input file. Parser performs multiple sanity checks after it finishes parsing to detect inappropriately specified components, e.g., the positions of two components are set such that they overlap in space. A *CTM module* takes the valid components set from the parser, divides them into fine-grained sub-components, and stores them into a spatial database. Next, the CTM module detects physical contacts among sub-components and builds a compact thermal model. Finally, the compact thermal model is given to a *Solver module*. The solver uses the thermal model along with the power trace coming from the parser to compute temperature maps of all components. The Solver applies a *parallel method* using GPUs to solve for temperature results more quickly when GPU hardware is available.

4. COMPACT THERMAL MODELING

There is a well-known duality between the thermal and electrical phenomena [21]. The compact thermal modeling methods build an equivalent RC circuit based on the original thermal system. In this paper, we focus on generating the steady-state temperature maps for components inside a smartphone because the objective of thermal design and management is to ensure that the device can run continuously without exceeding a given temperature threshold. Therefore, the device is modeled by using a thermal resistance network only.

To build a compact thermal model, Therminator divides specified components into *sub-components* with smaller dimensions and checks for physical contacts among sub-components. Finer granularity of sub-component division helps to produce more accurate temperature maps at the cost of increased runtime and memory usage. Each sub-component is modeled as a node in the thermal resistance network and has a single temperature value. A thermal resistance is calculated for every contacted sub-component

pairs, based on their material properties, dimensions, and relative positions.

Figure 2 shows a small part of thermal resistance network for the Qualcomm MSM8660 Mobile Developer Platform (MDP) [12]. The components in Figure 2, from top to bottom, include screen protector, display module, PCB and IC chips, battery, and rear case. Terminator breaks various components into non-equal number of sub-components according to their importance and requirements of solution quality. For two adjacent sub-components i and j, the thermal resistance is calculated by serially connecting two thermal resistors from their centers to the shared surface,

$$r(j,i) = r(i,j) = r_i + r_j = \frac{1}{A}\left(\frac{t_i}{k_i} + \frac{t_j}{k_j}\right) \quad (1)$$

where A is the common area between these two contacted sub-components, k_i and k_j are the thermal conductivity, and t_i and t_j are the perpendicular distances from the center of sub-components to the shared surface, respectively. Note that adjacencies between sub-components are detected in a 3D space and thereby, we account for orthotropism in the material thermal conductivity.

At the boundary of the device, heat diffuses to the ambient environment (air). Thus, the boundary thermal resistance between the i-th sub-component and the ambient air is calculated as,

$$r(i,amb) = r_i + r_{amb} = \frac{1}{A}\left(\frac{t_i}{k_i} + \frac{1}{h_{air}}\right) \quad (2)$$

where h_{air} is the air heat transfer coefficient. In the natural convection condition, h_{air} has the value of 5~25 $W/(m^2K)$ [22].

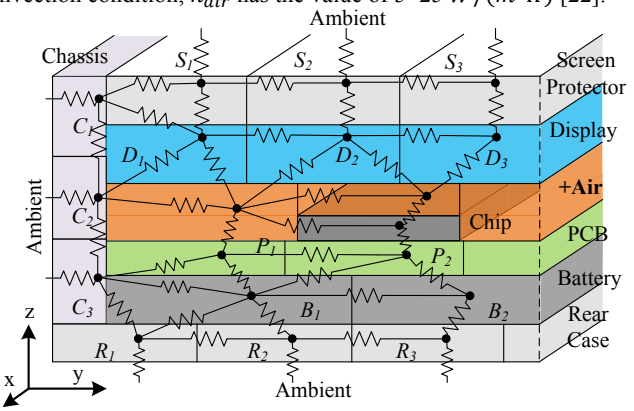

Figure 2. A cross-section view of the thermal resistance network in a simple smartphone model.

Note that empty spaces, shown as orange areas in Figure 2, are left in the design specifications. Ignoring these empty spaces, i.e., not calculating the thermal resistance between them and adjacent components will completely disable the heat flow through them and subsequently result in temperatures over-estimation. Thus, to avoid this issue, Therminator does *VoidFill* – i.e., it automatically identifies these empty spaces and fills them with air, as shown in Figure 2. Note that it is not practical to model the internal air using compact modeling of fluids in our problem, due to the lack of specific air circulation channels in smartphones. Therefore, in the steady-state, the air flow is ignored and the air is modeled like other sub-components. We apply a correction factor to the thermal conductivity of the air to account for this simplification.

Having built the resistance network, we obtain heat flow equations for all sub-components in a matrix format as follows,

$$\boldsymbol{G}\,\vec{\boldsymbol{T}} = \vec{\boldsymbol{P}} \quad (3)$$

where $\vec{\boldsymbol{T}}$ is the vector of all sub-component temperatures, $\boldsymbol{G}$ is the conductance matrix derived from the thermal resistance network, and $\vec{\boldsymbol{P}}$ is the heat generation vector, which includes the heat generation of sub-components and heat diffusion from the device to the ambient environment. Therminator adopts the *LUP decomposition* method to decompose $\boldsymbol{G}$ into a lower and upper triangular matrices, and then applies *forward and backward substitution* to solve for $\vec{\boldsymbol{T}}$. Advanced matrix solver libraries enabling GPU-acceleration are also included to reduce the runtime for fine-grained temperature maps.

5. THERMINATOR IMPLEMENTATION

Therminator is implemented using C++ and compiled by GCC 4.7. The parser adopts *PugiXML* [23], an open source, light-weight, and fast C++ XML processing library. The built-in material library is a class called `Materials` which holds default material properties and its data are updated by the parser. All components and sub-components are instances of `Component` and `Subcomponent` classes, respectively. A `Device` class keeps track of sub-components objects using a spatial database. Another class called `Model` takes the `device` object and builds the thermal model based on Equations (1) and (2). Several geometric utility methods are implemented in order to perform basic spatial queries on sub-components, e.g., checking the physical contact between every two sub-components, determining if they have overlap in space, and calculating their common area. Moreover, the `Model` class calls another parser to read the `power.trace` file which contains the power consumption of each component.

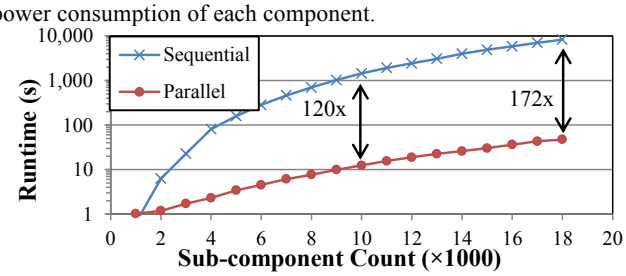

Figure 3. Comparison of runtime of sequential and parallel methods for different sub-component counts.

Matrix solving techniques, namely, the LUP decomposition method followed by the forward and backward substitution method, are implemented using the sequential method (which utilizes the CPU) and the parallel method (which utilizes the GPU), respectively. For the parallel method, Therminator adopts *CULA Dense* [24], which is a set of GPU-accelerated linear algebra libraries utilizing the *NVIDIA CUDA* parallel computing platform. One can observe that the parallel method speeds up Therminator by more than two orders of magnitude against the sequential method, as shown in Figure 3. Runtime results of both methods are measured on a server with 4×Intel Xeon E7-8837 CPUs, 64GB of memory, and an NVIDIA Quadro K5000 GPU.

6. THERMINATOR EVALUATION

6.1 Validation of the Therminator Results

We use a Qualcomm MSM8660 MDP [12] as the target system to validate Therminator results. The MSM8660 MDP has a dual-core 1.5GHz CPU, Adreno 220 GPU, 1GB LPDDR2 RAM, 3.61-inch touch screen, and a 1,300mAh Li-ion battery. A smartphone consists of a large number of small components with irregular geometric shapes and complicated material compositions. In this work, we try our best to identify the major components in the MSM8660 MDP and obtain the thermal properties of these components. Figure 4(a) shows a teardown of the MSM8660 MDP. We create a model for MSM8660 MDP device by identifying major components that have thermal impact to the entire device and measure their dimensions and relative positions. Components identified include rear case, chassis, battery, PCB, display, screen protector, and some ICs, such as AP, DRAM, eMMC, GPS and WiFi. The detailed material properties and dimensions for components are not shown due to the limited paper space. We draw the MSM8660 MDP model in Autodesk software, as shown in Figure 4(b), and perform CFD thermal analysis. We treat CFD results as golden results and compare Therminator results with

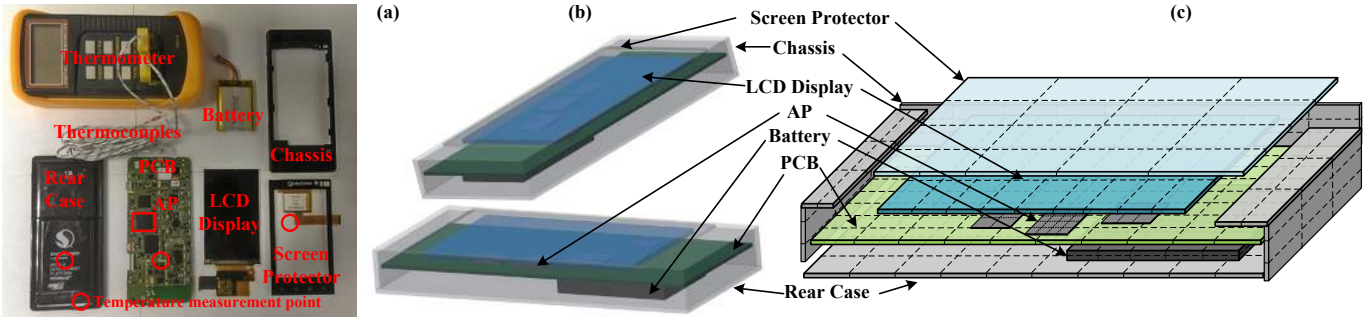

Figure 4. (a) Teardown of MSM8660 MDP device and temperature measurement kits (circle marks are temperature measurement points. Note for the PCB, thermocouple is attached onto the other side), (b) CFD drawing, and (c) Therminator 3-D visualization.

Table 1. Temperatures obtained from the thermocouple measurement (TCM), Autodesk Simulation CFD, and Therminator. Note the AP junction temperature is read from temperature register (Reg) instead of measurement. The ambient temperature is 23.0˚C.

Use Case	$T_{screen\ hot\ spot}$ (˚C)			$T_{rear\ case\ hot\ spot}$ (˚C)			$T_{PCB\ (near\ battery)}$ (˚C)			$T_{AP\ junction}$ (˚C)		
	TCM	CFD	Therminator	TCM	CFD	Therminator	TCM	CFD	Therminator	Reg	CFD	Therminator
StabilityTest	38.1	38.4	38.5	38.4	39.1	38.7	44.9	44.5	44.4	60	58.6	59.3
Candy Crush	37.2	37.8	37.7	38.4	39.2	38.9	46.2	44.6	44.8	59	59.0	59.5
YouTube	35.8	37.0	36.7	34.6	34.4	34.2	39.3	38.4	38.3	43	45.2	45.4
Camcorder	31.7	32.2	32.1	33.3	32.6	32.4	36.9	36.2	36.2	42	42.7	43.3
Video playback	30.2	30.8	30.7	30.5	30.8	30.7	33.3	33.4	33.4	39	39.4	40.0

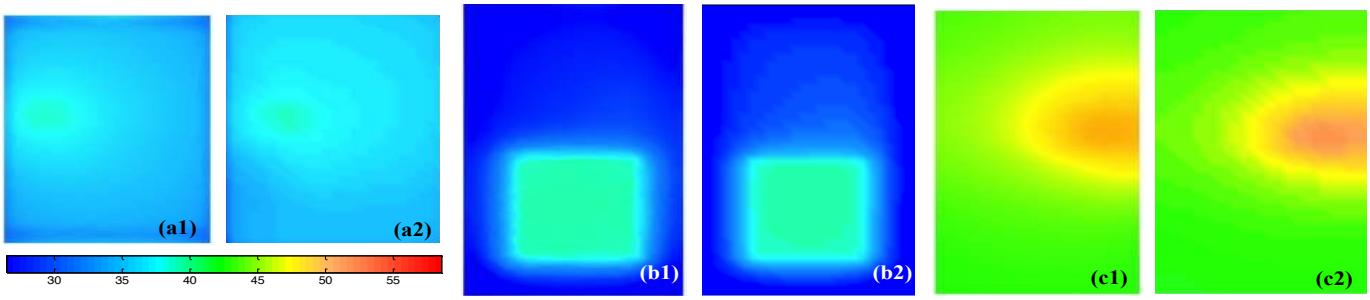

Figure 5. Temperature maps produced by Autodesk Simulation CFD (a1, b1, c1) and by Therminator (a2, b2, c2) for the screen protector (a), rear case (b), and PCB (c) for the StabilityTest use case.

them. Thus, a similar MDP device model, including the aforesaid components, their dimensions, relative positions and material properties, is specified in the *specs.xml* file for Therminator. Figure 4(c) visualizes the 3-D layout model that Therminator creates from the input file. Note that Therminator applies different granularity to different components.

We run a few representative use cases that utilize different components and consume various amounts of power. Use cases tested in this work are *StabilityTest* (an app that heavily stresses CPU and GPU [25]), casual gaming (*Candy Crush*), *YouTube* video streaming, camcorder (video recording), and a local video playback. We adopt *Trepn Profiler* [26] to record the per component power consumption breakdown of this device, and provide as inputs for both CFD simulation software and Therminator. Note that we assign the total power consumption of some small components (interconnects, sensors, etc.) to the PCB uniformly because we have no access to the schematic diagram of the MSM8660 MDP to precisely locate them.

We use thermocouples to measure temperatures at three locations in MSM8660 MDP, shown as red circles in Figure 4(a). We measure 1) hot spot on the screen right above the AP; 2) hot spot on the rear case below the battery (because there is a big air gap between PCB and rear case, the hot spot on the rear case is located below the battery); and 3) the PCB (the opposite side of the board shown in Figure 4(a).) The ambient temperature is measured as 23.0˚C during the experiments. We access *Sysfs* of the MDP device through the *Android Debug Bridge* interface and obtain the AP junction temperature by reading the temperature register in `/sys/class/thermal/thermal_zone2` directory. Note that the temperature register only has the accuracy of ±1˚C.

Table 1 compares temperature of aforementioned regions obtained through thermocouple measurements, CFD simulations, and Therminator. We first compare thermocouple measurement results and CFD simulation results. One can see that CFD simulation produces accurate results for all tested use cases and all regions. The maximum and average temperature error are 2.4˚C and 0.7˚C (11.0% and 4.7%), respectively. The error mainly comes from simplifications in modeling the real device and inaccuracies in determining component material properties. Note that the largest error (2.4˚C) comes from the AP junction temperature in YouTube use case. A potential reason might be the inaccuracy of the temperature register (i.e., ±1˚C).

Next, CFD results are used as golden results and we compare Therminator results with them. We divide specified components into 7,336 sub-components in total in Therminator. Table 1 shows that for all use cases and temperature points, the maximum and average errors of Therminator are only 0.7˚C and 0.25˚C (3.65% and 1.42%), respectively, compared to CFD results. Figure 5 shows more detailed comparisons of temperature maps, produced by CFD simulation and Therminator, of front screen, rear case, and PCB. One can see that Therminator is able to accurately capture not only the temperature of a particular hot spot, but also temperature maps of the entire smartphone device. Therefore, Therminator matches very well with the commercial CFD tool, given the same input models.

6.2 Convergence of the Therminator Results

Therminator can generate more detailed temperature maps at higher resolution with longer runtime. We study the convergence of temperature versus total the number of sub-components created by

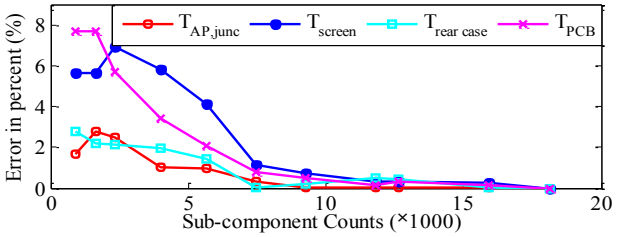

Figure 6. Therminator results convergence and runtime versus sub-component counts for the StabilityTest use case.

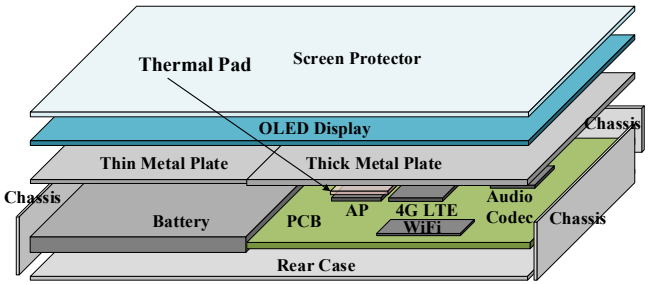

Figure 7. 3-D layout for Samsung Galaxy S4. Sub-components are not shown.

Therminator for MSM8660 MDP in Figure 6. We calculate *convergence errors* at different resolutions by comparing temperature results obtained at a particular resolution to those obtained at the highest resolution that we have tested (18,109 sub-components in total). One can see that the convergence errors of all four temperature points drop below 1% when the total sub-components number is above 7,000. According to results reported in Section 6.1, the difference of Therminator results compared to CFD results is only 1.42% for 7,500 sub-components. The runtime of Therminator at that resolution is less than seven seconds.

7. CASE STUDY

Therminator is versatile in handling different form-factor devices as long as input files are provided properly. In this section, we provide a case study targeted at Samsung Galaxy S4. Samsung Galaxy S4 is a flagship commercial smartphone released in 2013. Unlike the MSM8660 MDP device, Samsung Galaxy S4 does not provide power consumption due to some commercial reasons. Thus, the power consumption for major components, i.e., AP (CPU and GPU) and display, are estimated by measuring the total power consumption of Galaxy S4 at the battery output terminals and scaling them to the power breakdown ratio as reported in [27]. A simplified model of Galaxy S4 is also created, as shown in Figure 7. An AP floorplan describing locations of CPU and GPU is specified in the `specs.xml` file for better estimation accuracy.

We notice that in Galaxy S4, the thermal governor throttles the CPU, GPU, and memory operating frequency such that the skin temperature will not exceed 45˚C, i.e., the skin thermal governor has the temperature setpoint of 45˚C. The critical temperature of AP junction is usually quite high, say 85˚C, and thereby the frequency throttling we have observed is triggered by the skin thermal governor. We validate Therminator results for the maximum skin temperature located on the front screen (denoted as T_{skin}) and the AP junction temperature ($T_{AP,junc}$) against the thermocouple measurement results. The measurements results and Therminator results in the same condition of power consumption are underlined in Table 2. One can see that the temperature error produced by Therminator is within 0.5˚C (2%).

To simulate the effect of frequency throttling utilized by the thermal governor, we scale the total power consumption to produce different steady-state skin temperatures. Table 2 reports the corresponding T_{skin} and $T_{AP,junc}$ values for different AP power consumption values. To better study the effect of skin temperature

Table 2. Skin temperature and AP junction temperature obtained by thermocouple measurement (TCM) and Therminator at different AP power consumption levels.

Method	Temperature (˚C)		Power (W)		
	$T_{AP,junc}$	T_{skin}	P_{AP}*	$P_{AP,leak}$	$P_{AP,dyn}$
TCM	62.5	44.8	2.20	0.15	2.05
Therminator	68.0	47.7	2.64	0.18	2.46
	66.5	47.1	2.53	0.17	2.36
	65.1	46.5	2.42	0.16	2.26
	63.7	45.9	2.31	0.15	2.16
	62.3	45.3	2.20	0.15	2.05
	60.9	44.7	2.09	0.15	1.94
	59.4	44.1	1.98	0.13	1.85
	58.0	43.5	1.87	0.13	1.74
	56.1	42.9	1.76	0.12	1.64
	55.2	42.4	1.65	0.12	1.53
	53.8	41.8	1.54	0.11	1.43
	52.3	41.2	1.43	0.11	1.32
	50.9	40.6	1.32	0.11	1.21
	49.5	40.0	1.21	0.10	1.11
	48.1	39.4	1.10	0.10	1.00

* P_{AP} includes power consumption of both CPU and GPU.

on the device performance, we obtain the dynamic power consumption by subtracting the leakage power consumption, estimated by using McPAT [28], from the total AP power consumption values. Note that we use average AP temperature to estimate leakage power consumption values. Each row in Table 2 indicates a dynamic power consumption level when that specific skin temperature is met. In other words, when the skin thermal governor sets the target T_{skin} as the values listed in the third column of Table 2, the approximated AP's dynamic power consumption allotment are shown in the fifth column.

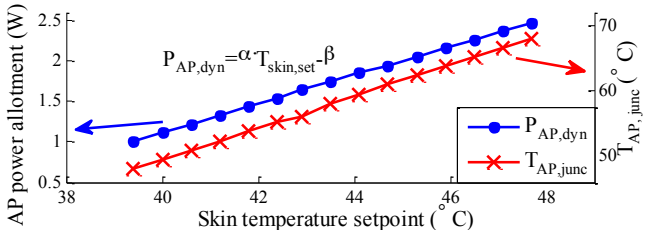

Figure 8. AP power consumption and junction temperature versus various skin temperature setpoints.

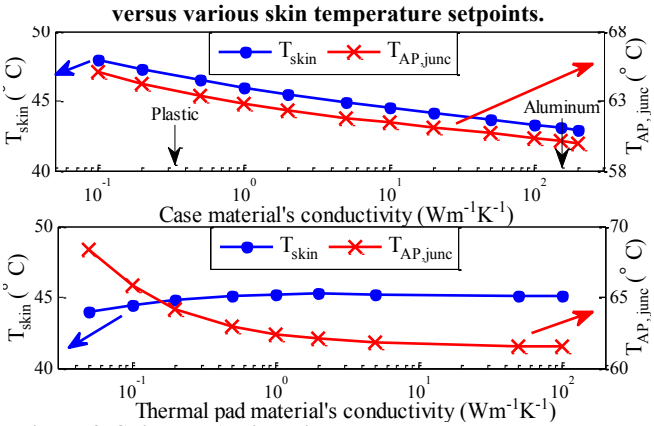

Figure 9. Skin and AP junction temperature versus rear case material (a) and thermal pad material (b) for $P_{AP} = 2.2$W.

Figure 8 plots the AP's dynamic power consumption allotment, denoted by $P_{AP,alt}$, versus the skin temperature setpoint, denoted by $T_{skin,set}$, as the latter is a typical variable in various thermal management policies. The blue dots indicates that $P_{AP,alt}$ (which is proportional to the device operating frequency and therefore, the

device performance) has a linear relationship with the setpoint value of skin temperature. From the data presented in Figure 8, we capture this relationship as,

$$P_{AP,alt} = \alpha \cdot T_{skin,set} - \beta \quad (4)$$

where $\alpha = 0.18$ W/K and $\beta = 5.92$ W. Since the device performance highly depends on $T_{skin,set}$, allowing high skin temperature results in significant performance improvement. For instance, increasing $T_{skin,set}$ from 45˚C to 48˚C results in 15.5% increase of $P_{AP,alt}$, i.e., an increase from 1.93W to 2.23W. On the other hand, decreasing $T_{skin,set}$ from 45˚C to 42˚C results a decrease from 1.93W to 1.63W. In addition, one can also observe from Figure 8 that the AP's junction temperature also linearly depends on the skin temperature setpoint (red crosses).

Clearly, modifying the thermal path design for a device affects its peak performance level. We study the thermal impact of thermal properties of the device exterior case by exploring its thermal conductivity from very low value (insulation material) to a high value (conductive material). Figure 9 (a) shows that both of T_{skin} and $T_{AP,junc}$ decrease when using higher thermal conductivity materials for the exterior case of the device. More precisely, adopting aluminum as the device case results in 2~3˚C lower T_{skin} and $T_{AP,junc}$, comparing with using pure plastic as the device case. This temperature reduction is helpful in improving the device performance. In practice, device manufacturers may also account for other factors such as the manufacturing cost.

We also investigate the impact of the material composition of the thermal pad, which is attached on top of the AP, and report the results in Figure 9 (b). A clear trade-off can be observed between T_{skin} and $T_{AP,junc}$ at various types of materials. This observation complies with results reported by a group of researchers at Texas Instrument [3]. The optimal thermal path design should touch the AP junction temperature constraint and skin temperature constraint at the same time. According to our study, from the thermal path design perspective, adopting a thermal pad with lower thermal conductivity on top of the AP achieves better performance. This is because T_{skin} is usually more critical in smartphones and a low thermal conductivity material hinders the heat flow to the device skin. However, in practice, some other factors (such as accelerated aging of AP and high leakage power at high temperatures) may prevent the usage of low thermal conductivity material.

8. CONCLUSION

We presented Therminator, a component-level compact-thermal-modeling-based thermal simulator targeting small form-factor devices in this work. Therminator is an early-stage, full-device thermal analyzer that produces accurate steady-state temperature maps of all components (ICs, boards, screens, cases, etc.) in a smartphone, from the application processor to the skin of device, with a fast runtime. Therminator provides great flexibility in handling different user-specified design specifications and use cases. We validated temperature results produced by Therminator against real temperature measurements using thermocouples and simulations using a commercial computational-fluid-dynamics tool on the Qualcomm MSM8660 MDP device. We also provided a case study on Samsung Galaxy S4 by using Therminator, showing that the device performance is linearly related to the device skin temperature. In addition, the impact of the thermal path design on the skin and AP junction temperature was also studied.

ACKNOWLEDGEMENT

This research is supported by grants from the PERFECT program of the Defense Advanced Research Projects Agency and the Software and Hardware Foundations of the National Science Foundation.

REFERENCES

[1] M. Pedram and S. Nazarian, "Thermal Modeling, Analysis, and Management in VLSI Circuits: Principles and Methods," *Proc. IEEE*, vol. 94, no. 8, pp. 1487–1501, 2006.
[2] J. Srinivasan *et al.*, "The case for lifetime reliability-aware microprocessors," in *ISCA*, 2004.
[3] S. P. Gurrum *et al.*, "Generic thermal analysis for phone and tablet systems," in *ECTC*, 2012, pp. 1488–1492.
[4] J. Rajmond and A. Fodor, "Thermal management of embedded devices," in *ISSE*, 2013, pp. 30–34.
[5] A. L. Shimpi, "The ARM vs x86 Wars Have Begun: In-Depth Power Analysis of Atom, Krait & Cortex A15." [Online]. Available: http://www.anandtech.com/show/6536/arm-vs-x86-the-real-showdown.
[6] E. Arens and H. Zhang, "The skin's role in human thermoregulation and comfort," *Indoor Environ. Qual.*, Oct. 2006.
[7] G. L. Wasner and J. A. Brock, "Determinants of thermal pain thresholds in normal subjects," *Clin. Neurophysiol. Off. J. Int. Fed. Clin. Neurophysiol.*, vol. 119, no. 10, pp. 2389–2395, Oct. 2008.
[8] A. Ku, "Asus Transformer Pad TF300T Review: Tegra 3, More Affordable - An Affordable Transformer Prime Derivative?," *Tom's Hardware*. [Online]. Available: http://www.tomshardware.com/reviews/transformer-pad-tf300t-tegra-3-benchmark-review,3179.html.
[9] J. A. Kaplan, "New Apple iPad hits 116 degrees, Consumer Reports says," *FoxNews.com*. [Online]. Available: http://www.foxnews.com/tech/2012/03/20/ipads-not-overheating-apple-says/.
[10] M.-N. Sabry, "Compact thermal models for electronic systems," *IEEE Trans. Compon. Packag. Technol.*, vol. 26, no. 1, pp. 179–185, 2003.
[11] K. Skadron et al., "Temperature-aware Microarchitecture: Modeling and Implementation," *ACM TACO*, no. 1, pp. 94–125, Mar. 2004.
[12] "Snapdragon MDP Mobile Development Platform - Legacy Devices," *Qualcomm Developer Network*. [Online]. Available: https://developer.qualcomm.com/mobile-development/development-devices-boards/mobile-development-devices/snapdragon-mdp-legacy-devices.
[13] "Simulation Software | Mechanical, CFD, Plastics | Autodesk." [Online]. Available: http://www.autodesk.com/products/autodesk-simulation-family/overview.
[14] Y. Han *et al.*, "Temptor: A Lightweight Runtime Temperature Monitoring Tool Using Performance Counters," in *TACS*, 2006.
[15] J. Meng *et al.*, "Optimizing Energy Efficiency of 3-D Multicore Systems with Stacked DRAM Under Power and Thermal Constraints," in *DAC*, 2012.
[16] M. J. Dousti and M. Pedram, "Platform-dependent, leakage-aware control of the driving current of embedded thermoelectric coolers," in *ISLPED*, 2013.
[17] A. Sridhar *et al.*, "3D-ICE: Fast Compact Transient Thermal Modeling for 3D ICs with Inter-tier Liquid Cooling," in *ICCAD*, 2010.
[18] Z. Luo *et al.*, "System thermal analysis for mobile phone," *Appl. Therm. Eng.*, vol. 28, no. 14–15, pp. 1889–1895, Oct. 2008.
[19] L. Zhang *et al.*, "Accurate online power estimation and automatic battery behavior based power model generation for smartphones," in *CODES+ISSS*, 2010.
[20] A. Pathak *et al.*, "Where is the Energy Spent Inside My App?: Fine Grained Energy Accounting on Smartphones with Eprof," in *EuroSys*, 2012.
[21] F. Kreith, *The CRC handbook of thermal engineering*. CRC Press, 2000.
[22] Y. A. Çengel, *Heat and mass transfer: a practical approach*. Boston: McGraw-Hill, 2007.
[23] "pugixml," *pugixml*. [Online]. Available: http://pugixml.org/.
[24] "CULA." [Online]. Available: http://www.culatools.com.
[25] "StabilityTest." [Online]. Available: https://play.google.com/store/apps/details?id=com.into.stability&hl=en.
[26] "Trepn Profiler," *Qualcomm Developer Network*. [Online]. Available: https://developer.qualcomm.com/mobile-development/performance-tools/trepn-profiler.
[27] X. Chen *et al.*, "How is Energy Consumed in Smartphone Display Applications?," in *HotMobile*, 2013, pp. 3:1–3:6.
[28] S. Li et al., "McPAT: An integrated power, area, and timing modeling framework for multicore and manycore architectures," in *MICRO*, 2009, pp. 469–480.