Therminator 2: A Fast Thermal Simulator for Portable Devices

Mohammad Javad Dousti, *Member, IEEE,* Qing Xie,
Mahdi Nazemi, *Student Member, IEEE,* and Massoud Pedram, *Fellow, IEEE*

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 2*, an early stage, fast, full-device thermal analyzer, which generates accurate transient- and steady-state temperature maps of an entire smartphone starting from the application processor and other key device components, extending to the skin of the device itself. Therminator 2 uses advanced numerical optimization techniques to perform steady-state simulations 1.6 times faster than the prior art technique and is capable of performing transient-state simulations in real-time and 1.25 times faster than the prior art method. The thermal analysis is sensitive to detailed device specifications (including its material composition and 3D layout) as well as different use cases (each case specifying the set of active device components and their activity levels.) Therminator 2 considers all major components within the device, builds a corresponding compact thermal model for each component and the whole device, and produces their transient- and steady-state temperature maps. Temperature results obtained by using Therminator 2 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 and Google Nexus 5. A case study on a Samsung Galaxy S4 using Therminator 2 is provided to relate the device performance to the skin temperature and investigate the thermal path design.

***Index Terms*—Thermal simulation, compact thermal modeling, CTM, electrical-thermal duality, portable devices, transient-state temperature, steady-state temperature, skin temperature.**

I. 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. 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 and 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 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.

Mortiz and Henriques [3] conduct an extensive study on the effect of temperature on human skin. The result of this research is summarized in Fig. 1. This figure shows that the required time to cause a burn (first or second degree) decays exponentially as the temperature increases. These data are collected for adults and it is expected that burn occurs faster for children and old people. It is also reported that the maximum safe temperature for human skin is about 45 °C. This threshold has also been confirmed by other researchers [4], [5].

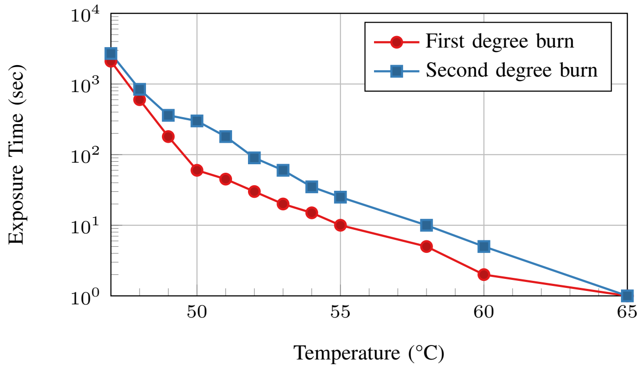


Fig. 1. Estimated exposure time of human skin to the hot water in order to result in a burn.

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, DSP, sometimes a baseband radio unit, and so on. The AP is a major heat generator in smartphones [6]. Due to the cost, form factor, noise, and safety issues, smartphones often 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 [6], [7]. Thermal management techniques, such as frequency throttling and voltage/frequency scaling, are also exploited to avoid high die temperatures. For instance, GSMArena studied Qualcomm Snapdragon 855 performance throttling in various Smartphone devices required to prevent the AP's junction temperature from exceeding an upper threshold [8]. Even with the aggressive performance throttling, maximum temperature of two of the devices under test, namely OnePlus 7 and Black Shark 2, reaches 47 °C. Authors claim devices in this temperature is "really uncomfortable" to be held in the hand.

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 threshold. Ideally, 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 [9]. It is reported that the hottest spot on Apple iPad 3 can reach as high as 47 °C while playing graphic intensive games [10]. 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 (i.e., temperature gradients) 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 fabrication. For instance, 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 threshold(s) is affected by how skin temperature depends on the CPU frequency.

Analyzing temperature maps at early stages of the design flow can significantly reduce the design time. Even though *computational fluid dynamics* (CFD) tools generate accurate temperature maps, they are slow, 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 [11], [12]. This method 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 2*, a CTM-based component-level thermal simulator targeting small form-factor mobile devices (such as smartphones and tablets). Therminator 2 (along with its predecessor [12]) is the first thermal simulator that targets small form-factor mobile devices (such as smartphones and tablets). 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 2 results have been validated against thermocouple measurements on a Qualcomm Mobile Developer Platform (MDP) [13] and simulation results generated by Autodesk Simulation CFD [14]. It 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). A detailed case study is conducted for Samsung Galaxy S4 by using Therminator 2. The temperature results relate the device performance to the device skin temperature, as well as the impact of the thermal path design. The new contributions of Therminator 2 compared to its predecessor [12] are listed below:

1) Proved that the conductance matrix is a sparse positive semi-definite matrix. This allowed us to employ faster numerical methods and exploit the power of parallel processing on multi-core CPUs to reduce the runtime by 33x on average for steady-state thermal simulations compared to the earlier version running on an NVIDIA GPU. Our suggested method is also shown to be on average 2.2x faster than HotSpot [15] and 1.6x faster than 3D-ICE [16].
2) Added the capability of fast transient-state thermal simulations. The accuracy of simulation results are verified through measurements performed on a Google Nexus 5 device.
3) Optimized the transient-state thermal simulation such that it can be done in real-time. This is a key benefit enabling Therminator 2 to be used along with a power model based on device components' activities (e.g., PowerTap [17]) in order to produce temperature maps while a device is being used without any thermocouple measurement or thermal imaging equipment. In order to achieve the speed up, first we show that the transient-state equation is a *stiff ordinary differential equation* (ODE). Next, we use an implicit Runge-Kutta method called *Rosenbrock* with adaptive steps to solve the stiff ODE. The faster steady-state calculation method (see item 1) is used in order to quickly calculate the initial solution for the ODE. We show that our transient-state solver is on-average 144x faster than the one employed by HotSpot and 1.25x faster than that of 3D-ICE.
4) Demonstrated that Therminator 2 steady- and transient-state solvers are scalable and can handle very large CTM problems with 1.68x less memory usage on average compared to 3D-ICE.

Therminator 2 is available as an open-source software at https://github.com/mjdousti/Therminator.

The rest of paper is organized as follows. Section II reviews related work. Next, Section III introduces the Therminator 2 software architecture. After that, the modeling methodology is explained in Section IV. Then, Section V describes techniques used to solve thermal equations. Next, Section VI elaborates implementation details and the evaluation of Therminator 2. A case study is provided in Section VII. Finally, Section VIII concludes the paper and presents the future work.

II. Related Work

HotSpot [15] is a successful early-stage CTM methodology targeting thermal analysis of a silicon die and its packaging which are cooled with a heat sink and possibly a fan. It generates die temperature maps. *Temptor* [18] is a tool based on HotSpot which allows the temperature prediction using performance counters instead of components' power trace.

Meng *et al.* [19] improved HotSpot by adding the 3D chip simulation support. *Teculator* [20] instruments HotSpot to support thermoelectric coolers. *3D-ICE* [16] is another thermal simulator targeting 3D 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) Device model has many more sub-components and solving relevant CTM equations require more efficient numerical techniques.
2) Multiple heat generators, including battery, display, and a number of IC chips.
3) Complex 3D layout where each component may be in vertical and/or horizontal contact with several other components.
4) Necessity of considering the internal air in the device.

Comparing to those tools, Therminator 2 focuses on component-level thermal modeling, in which the architecture-level details inside a single chip package are ignored (even though it has the capability of capturing chip-level floorplan.) We have shown in [17] that given the average power consumption of active components of the device every second, Therminator 2 can produce respective temperature maps at the same rate or faster. Last but not least, Section V demonstrates the performance advantage of Therminator 2 solver over HotSpot and 3D-ICE, especially for large CTM networks.

Several works have been conducted in studying the thermal design for smartphones and tablets. Luo *et al.* [21] established a simple thermal resistance network to analyze the whole mobile phone system. However, the thermal resistance network built in [21] is oversimplified as each component is modeled as one block with a single uniform temperature value. Gurrum *et al.* [6] modeled the smartphone in CFD tools and analyzed the thermal effect of using materials with different thermal conductivities through CFD simulation. Rajmond and Fodor [7] used CFD tools to show that attaching thermal pad on top of the AP significantly reduces the AP temperature. Egilmez *et al.* [22] suggested a method to estimate the mobile skin temperature using performance counters and temperature sensors. Using this method, they created a temperature governor for the Android operating system. Yun and Wu [23] created a model for a smartphone using a commercial CFD and shows the accuracy of the model through IR imaging. Kang *et al.* [24] studies the thermal implications of running popular applications on a smartphone using thermal imaging. Next, they suggest a simple linear model to predict the device skin temperature using internally available system statistics. Shen *et al.* [25] suggests an online power management technique based on Q-learning method which uses performance counters and temperature sensors. The model adapts itself based on the input in real-time. This model is capable of predicting temperature as well. Francesco and Rosing [26] suggest a linear model for predicting temperature of mobile devices based on the current reading of temperature and power. The system is numerically solved using N4SID method. Comparing to these tools, Therminator 2 focuses on component-level thermal modeling, in which the architecture-level details inside a single chip package are ignored. Besides, Therminator 2 can be used for realtime (online) simulations. We have shown in [17] that given the average power consumption of active components of the device every second, Therminator 2 can produce respective temperature maps at the same rate or faster. This eliminates the need for oversimplifying the thermal model into a simple linear equation which reduces the accuracy. Moreover, a full-device simulator enables device manufacturers to perform design space exploration (see Section VII). To the best of our knowledge, Therminator is the first tool targeting smartphones that automatically builds a compact thermal model from the device specification, and solves for temperature maps of all components accurately in real-time.

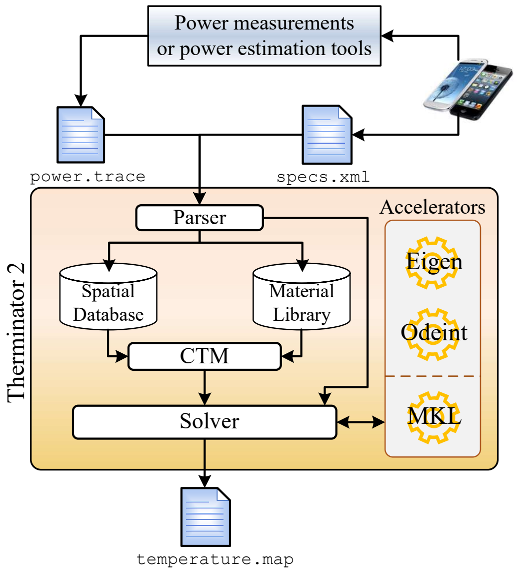


Fig. 2. Architecture of Therminator 2.

III. Therminator 2 Software Architecture

Fig. 2 depicts how Therminator 2 works. It takes two input files provided by a user. The `specs.xml` file describes the smartphone design, including components of interest, their geometric dimensions (length, width, and thickness), and relative positions. Users should provide properties of materials (i.e., thermal conductivity, density, and specific heat) used to manufacture the described device through this file. Listing 1 demonstrates a short `specs.xml`. The `power.trace` file provides the power consumption of active components that generate heat, e.g., ICs, the battery, and the display. The power trace of each component can be obtained through real measurements or other power estimation tools/methods such as [17], [27], [28]. `power.trace` is provided as a separate file so that one can easily interface a performance/power simulator (such as *GEM5*/*McPat* [28], [29]) with Therminator 2.

Therminator 2 has three main modules. A *parser module* parses input files, updates the *material library*, and makes a set comprised of components specified by the input file. Moreover, it performs multiple sanity checks to detect discrepancy among specified components, e.g., it identifies overlap of

```xml
1  <?xml version="1.0" encoding="UTF-8"?>
2  <device name="Device Name">
3    <!-- Device Aspect Ratio -->
4    <length>139.9e-3</length>
5    <width>71.2e-3</width>
6    <height>8.6e-3</height>
7    <!-- Ambient temperature in Kelvin -->
8    <temperature>298.15</temperature>
9    <power_trace_file>power.trace</power_trace_file>
10   <materials>
11     <material name="silicon">
12     <normal_conductivity>0.8</normal_conductivity>
13       <planar_conductivity>2</planar_conductivity>
14       <specific_heat>867</specific_heat>
15       <density>2330</density>
16     </material>
17   </materials>
18   <floorplans>
19     <floorplan name="Snapdragon_865">
20       <tile name="cpu-core1">
21         <length>2.8e-3</length>
22         <width>2.1e-3</width>
23         <x>5e-3</x>
24         <y>0.6e-3</y>
25       </tile>
26   </floorplans>
27   <components>
28     <component name="Snapdragon_865">
29       <material>silicon</material>
30       <power gen="yes">
31         <fill>no</fill> <!-- Fill the gap? -->
32       </power>
33       <lateral_connectivity>
34         yes
35       </lateral_connectivity>
36       <length>15.2e-3</length>
37       <width>15.1e-3</width>
38       <height>0.655e-3</height>
39       <x>102.39e-3</x>
40       <y>21.3e-3</y>
41       <z>4.56e-3</z>
42       <resolution>
43         <width>7</width>
44         <length>7</length>
45         <height>3</height>
46       </resolution>
47     </component>
48   </components>
49 </device>
```

Listing 1. An example `specs.xml` file. A complete file needs to list all components filling the device bounding box.

two components in space. A *CTM module* takes the validated component 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 benefits from various accelerators to quickly produce temperature maps.

IV. Compact Thermal Modeling

There is a well-known duality between the thermal and electrical phenomena [30]. This duality is summarized in Table I. The compact thermal modeling method builds an equivalent RC circuit based on the original thermal system.

TABLE I
Thermal Quantities and Their Electrical Duals

Thermal		**Electrical Dual**	
Thermal Quantity	**Unit**	**Quantity**	**Unit**
Temperature (T)	K	Voltage (V)	V
Power (P)	W	Current (I)	A
Thermal resistance (R_{th})	K/W	Electrical resistance (R)	Ω
Heat Capacity (C_{th})	J/K	Electrical capacitance (C)	F

To build a compact thermal model, Therminator 2 divides components (specified in `specs.xml`) 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 RC network and has a single temperature value. A thermal resistance is calculated for every sub-component pairs in contact, based on their material properties, dimensions, and the contact area. Similarly, a capacitance is added between every node and the ground. This capacitance captures the sub-component specific heat.

Fig. 3 shows a small part of a thermal RC network for the Qualcomm MSM8660 Mobile Developer Platform (MDP) (see [13] for details on MDP devices). The components in Fig. 3, from top to bottom, include screen protector, display module, PCB, IC chips, battery, and rear case. Therminator 2 breaks various components into non-equal number of sub-components according to their importance and requirements of solution quality (see lines 41 to 45 in Listing 1.) 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 as follows.

$$r_{i,j} = r_{j,i} = r_i + r_j = \frac{1}{A_{i,j}}\left(\frac{t_i}{g_i} + \frac{t_j}{g_j}\right), \tag{1}$$

where $A_{i,j}$ is the common area between these two contacted sub-components, g_i and g_j are thermal conductivities of their respective sub-components, and t_i and t_j are the perpendicular distances from the center of sub-components i and j to

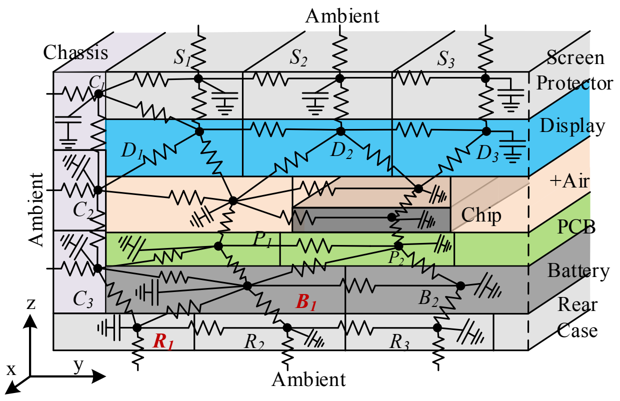


Fig. 3. A cross-section view of the thermal RC network in a simple smartphone model.

the shared surface, respectively. Note that adjacency between sub-components are detected in a 3D space and thereby, we account for *orthotropic thermal conductivity*. A material is orthotropic if its thermal conductivity varies in different directions. PCBs are a good examples of orthotropic materials; they have copper traces spanned in the horizontal plane and hence exhibit a higher thermal conductivity in that direction compared to the vertical direction.

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

$$r_{i,amb} = r_i + r_{amb} = \frac{1}{A_{i,amb}}\left(\frac{t_i}{g_i} + \frac{1}{h_{air}}\right), \tag{2}$$

where h_{air} is the air heat transfer coefficient. In the natural convection condition, h_{air} has the value of $5\sim25\,\mathrm{W/(m^2\cdot K)}$ [31].

Note that empty spaces, shown as orange areas in Fig. 3, are left in the design specifications. Ignoring these empty spaces, i.e., not calculating the thermal RC between them and adjacent components will completely disable the heat flow through them and subsequently result in temperature overestimation. Thus, to avoid this issue, they should be identified and filled with air, as shown in Fig. 3. Note that in our problem, due to the lack of specific air circulation channels in smartphones, it is not practical to model the internal air using compact modeling of fluids. Therefore, air flow is ignored and it is modeled like other solid sub-components. We apply a correction factor to the air thermal conductivity to account for this simplification.

The heat capacity assigned to the sub-component i, which is not at the boundary of the device, is calculated as

$$C_{th,i} = \gamma \cdot c_i \cdot \rho_i \cdot V_i, \tag{3}$$

where γ is a correction factor, whereas c_i, ρ_i, and V_i are the specific heat, the density, and the volume of sub-component i. γ is determined empirically as 0.5 to offset the effect of lumping capacitors. If the i^{th} component is at the boundary of the device, its heat capacity is calculated as

$$C^{bound}_{th,i} = \gamma\left(c_i \cdot \rho_i \cdot V_i + C^{conv}_{th,\square} \cdot A_{i,amb}\right), \tag{4}$$

where $C^{conv}_{th,\square}$ is the convection heat capacitance per unit area and $A_{i,amb}$ is the common area between the sub-component i and the ambient.

The ambience is modeled using a constant voltage source with the value of the ambient temperature. This voltage source is connected to the nodes corresponding to the sub-components at the boundary of the device. Power generation of components is modeled using current sources.

Solving the RC network made using the aforesaid technique gives the voltage at every node of the device which is equivalent to the temperature of sub-components in the initial device model. By nodal analysis, one can formulate this problem as

$$\boldsymbol{C}\frac{d\vec{T}(t)}{dt} + \boldsymbol{G}\vec{T}(t) = \vec{P}(t), \tag{5}$$

where t is time, $\boldsymbol{C}$ is the thermal capacitance matrix, $\boldsymbol{G}$ is the conductance matrix, $\vec{T}(t)$ is the temperature vector at time t, and $\vec{P}$ is the power consumption vector at time t. $\boldsymbol{C}$ is a diagonal matrix, i.e., each element on the main diagonal represents the heat capacity of its corresponding sub-component. $\boldsymbol{G}$ can be represented as

$$\boldsymbol{G} = \begin{bmatrix} \sum_{i\neq1} g_{1,i} & -g_{1,2} & \cdots & -g_{1,n} \\ -g_{2,1} & \sum_{i\neq2} g_{2,i} & \cdots & -g_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ -g_{n,1} & -g_{n,2} & \cdots & \sum_{i\neq n} g_{n,i} \end{bmatrix}, \tag{6}$$

where $g_{i,j}$ represents the thermal conductivity between sub-components i and j. Clearly, the following relation holds.

$$g_{i,j} = g_{j,i} = \frac{1}{r_{i,j}} = \frac{1}{r_{j,i}} \tag{7}$$

In the steady-state thermal analysis, similar to the DC analysis of RC circuits, the first term in Eq. (5) is dropped and the resultant system of linear equations is solved. On the other hand, for transient analysis, the entire Eq. (5), which forms a system of ordinary differential equations (ODEs), is solved. This is a nonhomogeneous, first-order linear system of ODEs, which has the following closed-form solution [32],

$$\vec{T}(t) = e^{-\boldsymbol{Y}t}\left(\int_{t_0}^{t} e^{\boldsymbol{Y}t}\boldsymbol{X}(t)dt + \vec{T}(t_0)\right), \tag{8}$$

where $\boldsymbol{X}(t)$=$\boldsymbol{C}^{-1}\vec{P}(t)$ and $\boldsymbol{Y}$=$\boldsymbol{C}^{-1}\boldsymbol{G}$. Calculating the numerical value of this solution is comprised of matrix exponentials and hence, requires numerical estimation of power series. Moreover, $\vec{P}(t)$ is given over some period of time (say from t to $t+\Delta t$) and hence, the integral is converted into a summation. As a result, Eq. (8) should be solved using numerical methods. In the next section, we investigate various techniques to quickly solve Eq. (5).

V. The Solver

A key issue in solving an ODE is determining an initial solution for it, which is denoted by $\vec{T}(t_0)$ in Eq. (8). To find such a solution, one can solve Eq. (5) in steady-state. Consequently, generating online temperature maps of a system consists of first solving the steady-state equation and the ODE afterwards.

A. Steady-State Analysis

In this section, we investigate two techniques for solving the system of linear equations presented in Eq. (5). The first technique is *LUP decomposition*, which is used in the initial version of Therminator, and the second method is sparse *Cholesky* decomposition, which is much faster and adopted in Therminator 2.

1) LUP Decomposition: The *LUP decomposition* method decomposes $\boldsymbol{G}$ into a lower and an upper triangular matrices, and then applies *forward and backward substitution* to solve Eq. (5) for $\vec{T}(t)$. Therminator 1 uses advanced matrix solver libraries enabling GPU-acceleration to reduce the runtime for fine-grained temperature maps.

2) Cholesky Decomposition: In order to speed up steady-state thermal simulations, first it is shown that the conductance matrix is a *positive semi-definite* (PSD) matrix. The definition of PSD matrices and the proof of the claim follow.

Definition 1. *Matrix $\boldsymbol{M}$ is positive definite (PD) if it is symmetric and $\vec{z}^\top \boldsymbol{M}\vec{z}$ is positive for any non-zero column vector $\vec{z}$ of real numbers. The definition of a positive semidefinite (PSD) matrix is the same except the fact that $\vec{z}^\top \boldsymbol{M}\vec{z}$ should be non-negative.*

Theorem 1. *The conductance matrix $\boldsymbol{G}$ is PSD.*

Proof. By definition, the conductance matrix is symmetric. Also note that thermal conductivities are positive values and $g_{i,j} \geq 0$. Now, consider $\vec{z}$ as a column vector where its i^{th} element is denoted by z_i. Calculating the value of $\vec{z}^\top \boldsymbol{G}\vec{z}$ gives

$$\begin{aligned}\vec{z}^\top \boldsymbol{G}\vec{z} &= \sum_i z_i^2\Big(\sum_{\substack{j\\ j\neq i}} g_{i,j}\Big) - \sum_{\substack{i,j\\ i\neq j}} z_i z_j g_{i,j}\\ &= \sum_{\substack{i,j\\ i\neq j}} z_i^2 g_{i,j} - z_i z_j g_{i,j}\\ &= \frac{1}{2}\sum_{\substack{i,j\\ i\neq j}} (z_i^2 g_{i,j} + z_j^2 g_{j,i}) - 2z_i z_j g_{i,j}\\ &= \frac{1}{2}\sum_{\substack{i,j\\ i\neq j}} (z_i^2 + z_j^2) g_{i,j} - 2z_i z_j g_{i,j}\\ &= \frac{1}{2}\sum_{\substack{i,j\\ i\neq j}} (z_i - z_j)^2 g_{i,j} \geq 0.\end{aligned}$$

Hence, $\boldsymbol{G}$ is PSD. □

Next, the conductance matrix is transformed into a *positive definite matrix*, which allows *Cholesky decomposition* to be used in order to solve Eq. (5). Note that the LUP decomposition is a generic matrix decomposition technique that can be applied to any matrix. Therminator 2 uses the Cholesky decomposition which is proven to be much faster than LUP decomposition [33].

In order to apply the Cholesky decomposition on a matrix, the matrix should be PD. The technique explained in [33] is employed in order to apply the Cholesky decomposition technique to a PSD matrix as explained next. Consider a matrix called $\boldsymbol{G_k} = \boldsymbol{G} + (1/k)\boldsymbol{I}$, where $\boldsymbol{I}$ is the identity matrix and k is a large positive value. Using the same argument presented in the proof of Theorem 1, it can be seen that $\boldsymbol{G_k}$ is PD. Moreover, the value of k can be chosen arbitrarily large such that $\boldsymbol{G_k}$ becomes an approximation of $\boldsymbol{G}$. As a result, it can be seen that matrix $\boldsymbol{G_k}$ can be decomposed to $\boldsymbol{L_k}\boldsymbol{L_k}^\top$, where $\boldsymbol{L_k}$ is a lower triangular matrix. As k approaches to infinity, $\boldsymbol{L_k}\boldsymbol{L_k}^\top$ tends to $\boldsymbol{L}\boldsymbol{L}^\top$, which is the decomposition of matrix $\boldsymbol{G}$. A very large k is chosen to calculate the Cholesky decomposition of $\boldsymbol{G}$ numerically with negligible loss of accuracy (less than 10^{-4} °C).

Moreover, it is shown that the conductance matrix is sparse and hence the sparse variant of Cholesky decomposition may be used to achieve further speed-up. The definition of a sparse matrix and the proof of the claim are as follows.

TABLE II
COMPARING ODE TECHNIQUES IN NUMERICALLY SOLVING EQ. 8

Type	Method Name	Order	Error Estimation	Step Count	Runtime (in sec)
Explicit	Explicit Euler	1	No	100k[†]	2.76
	Midpoint	2	No	100k[†]	8.32
	Runge-Kutta	4	No	100k[†]	11.06
	Cash-Karp	5	Yes	1849	0.40
	Dormand-Prince	5	Yes	2092	0.46
	Fehlberg	8	Yes	1364	0.89
	Adams-Bashforth	4[*]	No	100k[†]	2.81
	Adams-Bashforth-Moulton	4[*]	No	100k[†]	5.60
	Bulirsch-Stoer	variable	Yes	1102	0.66
Implicit	Rosenbrock	4	Yes	8	0.09

[*] This method takes the order as an input argument. We tried all possible values and chose the one with the fastest runtime which was 4.
[†] 100k is the maximum number of allowed steps.

Definition 2. *An $n \times n$ matrix is sparse, if at most $\mathcal{O}(n)$ of its elements are non-zero [34].*

Theorem 2. *The conductance matrix $\boldsymbol{G}$ is sparse.*

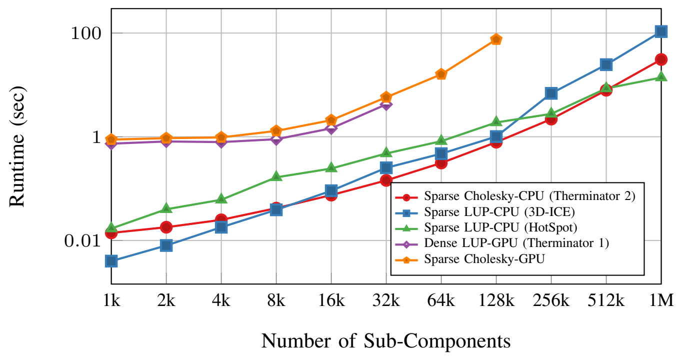
Proof. Every row of the conductance matrix corresponds to a sub-component in the CTM. When the sub-components are small enough, they have only six neighbors (i.e., only a sub-component resides at each side of a rectangular cuboid.) Considering the diagonal elements, there are seven non-zero elements in every row of the conductance matrix. Hence, the conductance matrix has a total of $7n = \mathcal{O}(n)$ non-zero elements. □

B. Transient Analysis

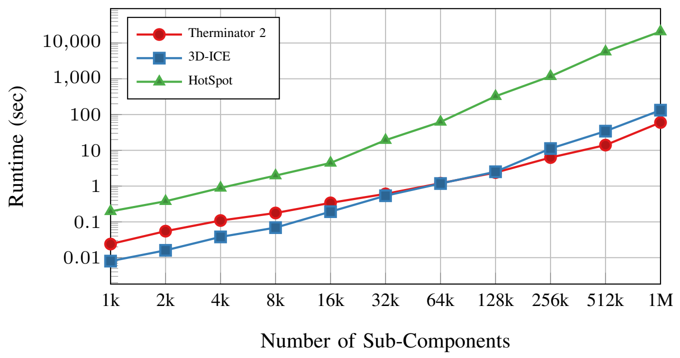
ODE solvers need to numerically calculate $\frac{d\vec{T}}{dt}$ many times and hence it is critical to ensure that it can be done quickly. Using Eq. 5 and notations in Eq. 8, one can write $\frac{d\vec{T}}{dt}$ as

$$\frac{d\vec{T}(t)}{dt} = \boldsymbol{C}^{-1}\vec{P}(t) - \boldsymbol{C}^{-1}\boldsymbol{G}\vec{T}(t) = \boldsymbol{X}(t) - \boldsymbol{Y}\vec{T}(t). \quad (9)$$

To efficiently calculate this, three avenues are taken. First, $\boldsymbol{Y}$ is pre-calculated and $\boldsymbol{X}$ is only calculated when $\vec{P}(t)$ changes. Next, it is noted that $\boldsymbol{Y}$ is sparse. The reason is as follows. Remember the fact that $\boldsymbol{G}$ was proven to be sparse. $\boldsymbol{Y}$ is formed by multiplying a diagonal matrix (i.e., $\boldsymbol{C}^{-1}$) to $\boldsymbol{G}$ and hence the resultant remains sparse. This sparsity reduces the order of computations from quadratic to linear. Next, we consider ten widely used explicit and implicit ODE solving techniques and compare their runtime. Table II summarizes the results of this comparison when solving the ODE for a Google Nexus 5 device with 4000 sub-components and $\Delta T = 1$s. In our experiments, we found that using adaptive step selection technique not only results in lower runtime and fewer solver iterations, but also does not require us to manually tune the step size. Hence, all reported runtimes are for methods with adaptive steps. Moreover, it is important to note that all approaches yield identical results when they are rounded to 0.1 °C. Consequently, for methods which allowed



(a) Steady-state simulation



(b) Transient-state simulation

Fig. 4. Runtime comparison for different number of sub-components.

error estimation (see Table II), we used 0.1 °C as the desired level of error. Using smaller error value would not be of practical use and is beyond the precision of our thermocouples which we used to verify the accuracy of Therminator 2 (see Section VI.)

As can be seen, explicit methods are slower than the implicit method and require more than two orders of magnitude steps to solve the ODE equation. Hence, one can conclude the fact that Eq. 8 is *stiff*. Explicit methods cannot handle Stiff equations properly. They are often numerically unstable unless the step size in solving the ODE is selected to be extremely small [33]. Choosing small steps in turn leads to more steps required to solve the problem and hence higher runtime as already been demonstrated in Table II. Consequently, we use *Rosenbrock* in Therminator 2 to solve the transient-state equation. Rosenbrock is a fourth-order implicit Runge-Kutta method and is detailed in [35].

C. Performance Benchmark

In this section, we benchmark the performance of Therminator 2 solver against its earlier version and prior art. These benchmarks are done for a Google Nexus 5 device model with different number of sub-components. CPU benchmarks are performed on an Intel Core i7-9750H CPU with a base frequency of 2.6GHz. This CPU is equipped with 6 cores and 12 threads and installed on a system with 32GB DDR4 memory. On the other hand, GPU benchmarks use an NVIDIA Titan RTX GPU with 24GB of memory. At the time of writing, this GPU is the highest-end consumer GPU available. In order to ensure the best performance from the NVIDIA GPU, *NVIDIA cuSOLVER* [36] is employed. NVIDIA cuSOLVER is a set of GPU-accelerated linear algebra libraries which utilize *NVIDIA CUDA* parallel computing platform. Note that all numerical methods presented in this section achieve identical results when they are rounded to 0.1 °C.

1) Steady state: In this part, we compare five methods explained earlier as follows:

- Sparse Cholesky-CPU (Therminator 2): This is the method we chose for Therminator 2. It employs Eigen library backed by Intel MKL PARDISO library [37].
- Sparse LUP (3D-ICE): 3D-ICE employs a library called SuperLU [38] which is backed by Intel MKL library.
- Cholesky-CPU (HotSpot): This method is employed by HotSpot 6 [39]. We enabled Intel MKL library [40] as suggested by the software user manual to achieve the peak performance it can offer.
- LUP-GPU (Therminator 1): This method is employed by Therminator 1.
- Sparse Cholesky-GPU: Similar to Therminator 2 method except this method is run on a GPU.

Fig. 4(a) compares all approaches explained above. Note that the figure is a log-log plot. One can easily see that CPU accelerated methods are much faster compared to their GPU counterparts. Moreover, GPU accelerated methods fail to scale to large sub-component counts. This is due to three reasons: First, initializing a GPU and copying the matrix and vector (i.e., G and $\vec{P}$ in Eq.5, respectively) from system memory to GPU memory has a non-trivial amount of overhead. Second, GPUs are designed to effectively perform batches of matrix computations, whereas in this problem we are interested in only solving a single linear equation. Last but not least, Intel CPUs benefit from *Single Instruction Multiple Data* (SIMD) operations, which are marketed as *Advanced Vector Extensions* (AVX) [41] instructions. These operations help implementations based on Intel CPUs achieve superior result. Aside from wide availability of Intel CPUs, it is important to note that the NVIDIA Titan RTX GPU is priced at $2500, whereas Intel Core i7-9750H has a retail price of $395.

As we expected, Fig. 4(a) shows that Cholesky decomposition is faster than LUP. On GPU, sparse-Cholesky decomposition can scale to larger sub-component counts compared with dense LUP; however, it still fails to scale to sub-component counts greater than 128k due to memory issues. We think that this is a limitation of NVIDIA cuSOLVER library otherwise the GPU memory should be able to accommodate matrices of that size. Unfortunately, NVIDIA cuSOLVER is not open source so we cannot verify the details. 3D-ICE performs the best for small problem sizes (smaller or equal to 8k) but from that point on, Therminator 2 takes the lead. Surprisingly, HotSpot only gains the best result for the largest problem size, i.e., 1M. On average, Therminator 2 performs 1.6x, 2.2x, and 33.2x better than 3D-ICE, HotSpot, and Therminator 1, respectively.

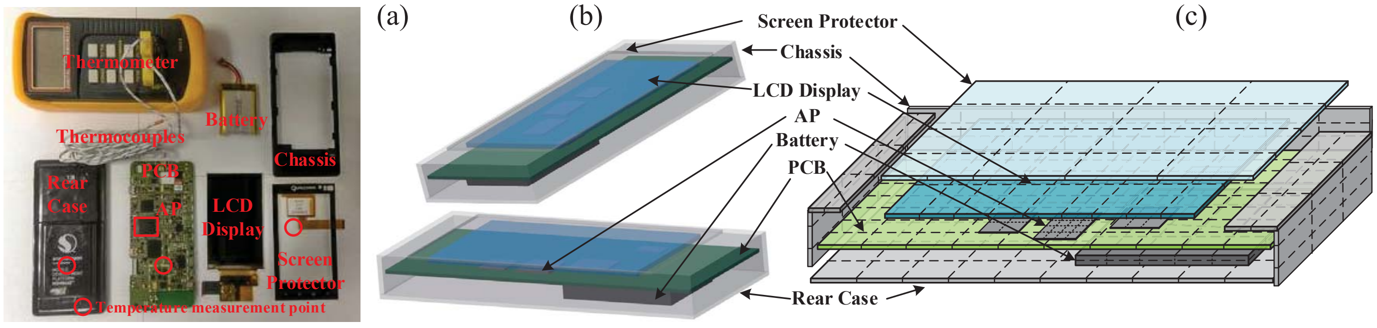


Fig. 5. (a) Teardown of MSM8660 MDP device and temperature measurement kits (circle marks are temperature measurement points. Note for the PCB, thermocouple is attached onto the other side), (b) CFD drawing, and (c) 3D visualization of Therminator 2 model.

2) Transient state: The initial version of Therminator does not support transient-state thermal analysis. However, HotSpot comes with a transient thermal simulator which is based on a 4th-order Runge-Kutta method with adaptive steps. Moreover, 3D-ICE also supports transient thermal simulation which is based on an implicit method. Similar to the previous part, HotSpot 6 is compiled with Intel MKL library and 3D-ICE is compiled with SuperLU. Fig. 4(b) summarizes the benchmarking results. We chose to perform the simulation for one second. Note that the figure is a log-log plot.

One can simply see Therminator 2 is significantly faster than HotSpot and surpasses 3D-ICE when the sub-component count goes beyond 64k. More concretely, Therminator 2 is on average 143.7x faster than HotSpot and 1.6x faster than 3D-ICE. For larger problem sizes, the speed up reaches 689.6x when compared to HotSpot and 4.4x when compared to 3D-ICE. To put this in perspective, it takes about 5.8 hours, 2.2 minutes, and 1 minute for HotSpot, 3D-ICE, and Therminator 2, respectively, to solve an ODE for one million sub-components. Also note that Therminator 2, while being faster, uses 3.8GB of system memory for 1M sub-components, whereas 3D-ICE uses 6.4GB of memory which is 1.68x higher.[1]

Comparing the high runtime of Runge-Kutta (which is an explicit method employed by HotSpot) with that of Rosenbrock (which is an implicit method used in Therminator 2) from Table II partly explains why Therminator 2 performs better. On the other hand, 3D-ICE utilizes an implicit method and hence has a performance close to that of Therminator 2. Besides the used ODE solving method, the speed up techniques we explained earlier for efficiently calculating $\frac{d\vec{T}}{dt}$ has helped Therminator 2 significantly.

VI. Implementation & Evaluation

A. Implementation

Therminator 2 is implemented using C++ and compiled by GCC 10.2. The parser adopts *PugiXML* [43], 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-component objects using a spatial database. Another class called `Model` takes the `device` object and builds the thermal model based on Equations (1)-(4). 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.

As explained previously, Therminator 2 requires to exploit the system's maximum performance. Hence, C++ is selected for its implementation. Besides, Therminator 2 uses *Eigen* [44] with *Intel Math Kernel Library* (MKL) [40] as a back-end to solve steady-state thermal equations. Eigen is an open-source high-performance high-level linear-algebra template library for C++, whereas MKL is a parallel low-level linear algebra library carefully-tuned for Intel CPUs. In order to solve transient-state thermal equations, Odeint Rosenbrock solver [45] was modified to use Eigen's sparse matrix representation with MKL as a back-end. Odeint is an open-source fast C++ library to solve ODEs.

B. Evaluation

1) Validation of Therminator 2 Results: A Qualcomm MSM8660 MDP [13] is used as the target system to validate Therminator 2 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, the major components of the MSM8660 MDP are identified and their thermal properties are obtained mostly from Autodesk Material Library [14]. Fig. 5(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,

[1]We used GNU Time [42] to measure the peak memory usag of each program. Note that GNU Time is different and more capable than the regular UNIX `time` command.

eMMC, GPS and WiFi. The detailed material properties and dimensions for components are not shown for brevity. The MSM8660 MDP model is drawn in the Autodesk software, as shown in Fig. 5(b), and the CFD thermal analysis is performed accordingly. CFD results are treated as golden results and they are compared to Therminator 2 results. 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 2. Fig. 5(c) visualizes the 3D layout model that Therminator 2 creates from the input file. Note that Therminator 2 applies different granularity to various components.

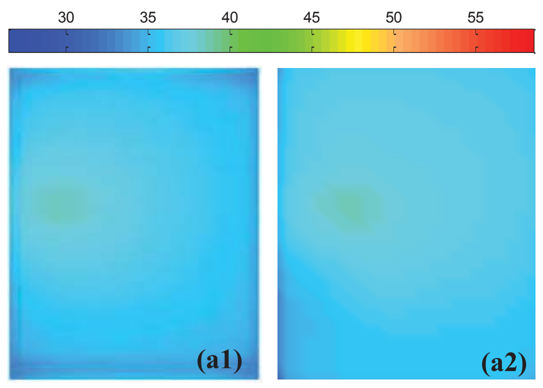
A few representative use cases are executed that utilize different components and consume various amounts of power. Use cases tested in this work are *StabilityTest* [46] (an application that heavily stresses CPU, GPU, and the memory), *Candy Crush* [47] (a popular mobile game), *YouTube* [48] (a famous video streaming application), the built-in camera application, and a local video playback. *Trepn Profiler* [49] is adopted to record the per-component power consumption breakdown of this device, and provide the results as inputs to both CFD simulation software and Therminator 2. Note that the total power consumption of some small components (interconnects, sensors, etc.) is assigned to the PCB uniformly because the schematic diagram of the MSM8660 MDP is not available to precisely locate them.

Temperatures of three different locations in MSM8660 MDP, shown as red circles in Fig. 5(a), are measured:

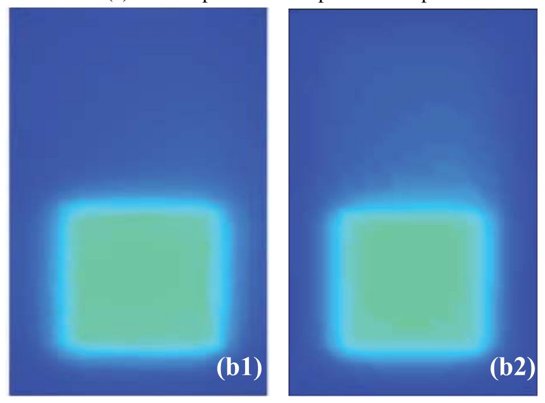
1) The hot spot on the screen located above the AP;
2) The hot spot on the rear case located below the battery (because there is a big air gap between the PCB and the rear case, the hot spot on the rear case is located below the battery); and
3) The PCB (the opposite side of the board shown in Fig. 5(a).)

Sysfs [50] of the MDP device is accessed through the *Android Debug Bridge* (ADB) interface and the AP junction temperature is obtained by reading the temperature register in the `/sys/class/thermal/thermal_zone2` directory. Note that the temperature register has the accuracy of ±1 °C. MCC USB-2408 [51] with type T thermocouple [51] is used to measure the temperature of hot spots on the rear case and PCB. The ambient temperature is also measured as 23.0 °C during the experiments. The type T thermocould has the accuracy of about ±0.56 °C.

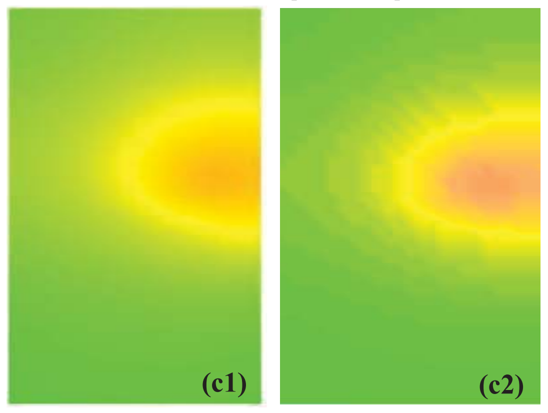
Table III compares the temperature of aforementioned regions obtained through thermocouple measurements, CFD simulations, and Therminator 2. First, the thermocouple measurement results and CFD simulation results are compared. One can see that CFD simulation produces accurate results for all tested use 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 the YouTube use case. A potential reason might be the inaccuracy of the temperature register (i.e., ±1 °C).



(a) Screen protector temperature maps



(b) Rear case temperature maps



(c) PCB temperature maps

Fig. 6. (a1, b1, c1) Temperature maps produced by Autodesk Simulation CFD and (a2, b2, c2) by Therminator 2 for the StabilityTest use case.

Next, CFD results are used as golden results and Therminator 2 results are compared with them. Specified components are divided into a total of 7,336 sub-components in Therminator 2. Table III shows that for all use cases and temperature points, the maximum and average errors of Therminator 2 are only 0.7 °C and 0.25 °C (3.65% and 1.42%), respectively, compared to CFD results. Fig. 6 shows more detailed comparisons of temperature maps, produced by the CFD simulation and Therminator 2, of the front screen, the rear case, and the PCB. One can see that Therminator 2 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 2 matches very well with the

TABLE III
TEMPERATURES OBTAINED FROM THE THERMOCOUPLE MEASUREMENT (TCM), AUTODESK SIMULATION CFD, AND THERMINATOR 2. NOTE THE AP JUNCTION TEMPERATURE IS READ FROM TEMPERATURE REGISTER (REG) INSTEAD OF DIRECT 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 2	TCM	CFD	Therminator 2	TCM	CFD	Therminator 2	Reg	CFD	Therminator 2
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

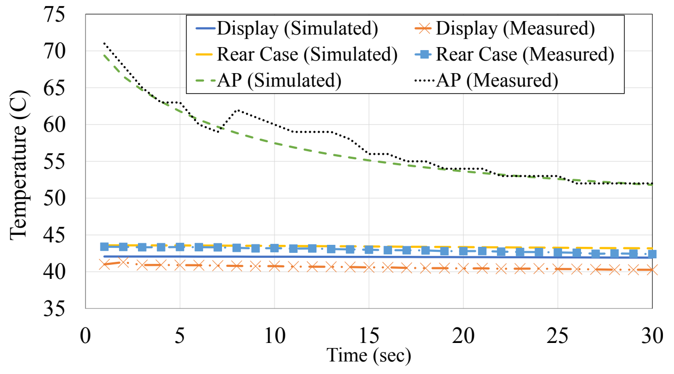


Fig. 7. Comparison of measured and simulated temperatures.

commercial CFD tool, given the same input models.

Also, a Google Nexus 5 smartphone is torn apart and its physical model is built. Next, temperature of three points is used to verify the transient-state simulation results: the AP internal temperature sensor and two sensors placed on the hottest spots of the rear case and the display of the phone. Similar to the previous experiment, MCC USB-2408 was used to log temperatures of these two sensors. StabilityTest [46] is executed to stress the AP, GPU and the memory of the smartphone and then the application is closed. Fig. 7 shows the transient temperature change when the smartphone is cooling down. On average, an error of 1.5°C, 1°C. and 0.5°C for the AP, display, and rear case were observed, respectively. Given the fact that the accuracy of the AP sensor and USB-2408 are ±1°C and ±0.56°C, respectively, the above error values are acceptable. Note that the AP temperature changes very quickly; however, the temperatures of display and rear case are reducing very slowly. This is due to the fact that AP has a higher temperature difference with the ambient and the thermal constant of AP is smaller than that of the display and rear case.

2) Convergence of Therminator 2 Results: Therminator 2 can generate more detailed temperature maps at higher resolution with slightly longer runtime. The convergence of temperature versus the total number of sub-components created by Therminator 2 is studied for MSM8660 MDP in Fig. 8. *Convergence errors* is calculated at different resolutions by comparing temperature results obtained at a particular resolution to those obtained at the highest resolution that are 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 the previously reported results, the difference of Therminator 2 results compared to CFD results is only 1.42% for 7,500 sub-components. The runtime of Therminator 2 at that resolution is less than 80 milliseconds for steady-state simulations.

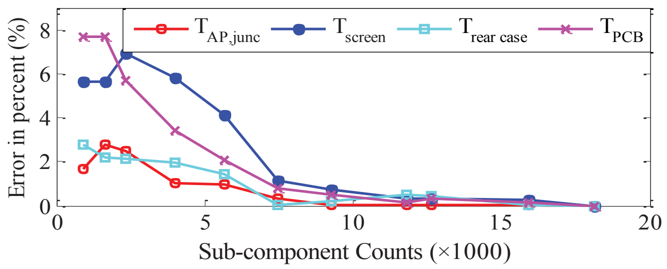


Fig. 8. Therminator 2 results convergence and runtime versus sub-component counts for the StabilityTest use case.

VII. CASE STUDY

Therminator 2 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. Unlike the MSM8660 MDP device, Samsung Galaxy S4 does not provide power consumption values 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 [52]. A simplified model of Galaxy S4 is also created, as shown in Fig. 9. An AP floorplan describing locations of CPU and GPU is specified in the `specs.xml` file to increase the accuracy of results.

We notice that in Galaxy S4, the thermal governor throttles the CPU, GPU, and memory operating frequencies 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 2 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 2 results in the same condition of power consumption are highlighted in Table IV. One can see that the temperature error produced by Therminator 2 (the shaded row) is within 0.5 °C (2%).

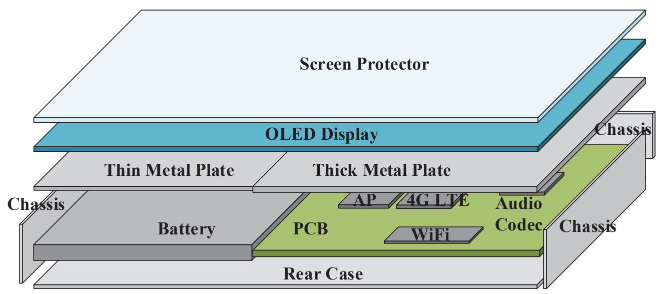


Fig. 9. 3D layout for Samsung Galaxy S4. Sub-components are not shown.

TABLE IV
SKIN TEMPERATURE AND AP JUNCTION TEMPERATURE OBTAINED BY THERMOCOUPLE MEASUREMENT (TCM) AND THERMINATOR 2 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 2	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

To simulate the effect of frequency throttling utilized by the thermal governor, the total power consumption is scaled to produce different steady-state skin temperatures. Table IV reports the corresponding T_{skin} and $T_{AP,junc}$ values for various AP power consumption values. To better study the effect of skin temperature on the device performance, the dynamic power consumption is obtained by subtracting the leakage power consumption, estimated by McPAT [28], from the total AP power consumption values. Note that the average AP temperature is used to estimate leakage power consumption values. Each row in Table IV 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 IV, the approximated AP's dynamic power consumption allotment are shown in the fifth column.

Fig. 10 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 Fig. 10, we capture this relationship as,

$$P_{AP,alt} = \alpha \cdot T_{skin,set} - \beta, \tag{10}$$

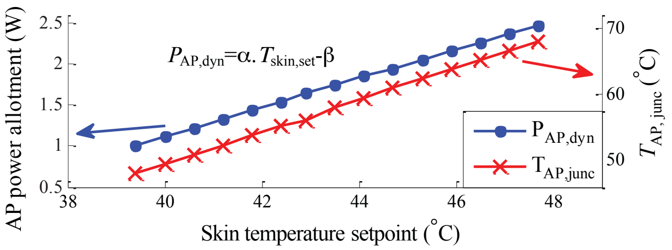


Fig. 10. AP power consumption and junction temperature versus various skin temperature setpoints.

where $\alpha = 0.18\,\text{W/K}$ and $\beta = 5.92\,\text{W}$. 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 1.93 W to 2.23 W. On the other hand, decreasing $T_{skin,set}$ from 45 °C to 42 °C results a decrease from 1.93 W to 1.63 W. In addition, one can also observe from Fig. 10 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. The impact of thermal properties of the device exterior case is studied by exploring its thermal conductivity from a low value (insulation material) to a high value (conductive material). Fig. 11(a) shows that both of T_{skin} and $T_{AP,junc}$ decrease when higher thermal conductivity materials are used for the exterior case of the device. More precisely, adopting aluminum as the device case results in 0.5 °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.

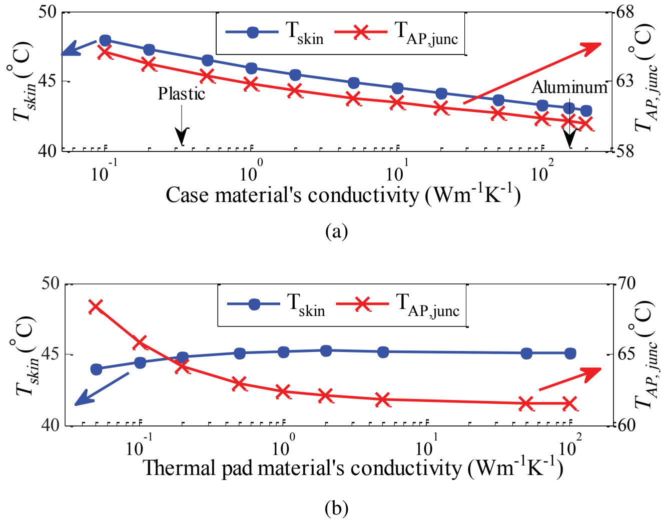


Fig. 11. (a) Skin and AP junction temperature versus rear case material and (b) thermal pad material for P_{AP} =2.2 W.

We also investigate the impact of the material composition of thermal pad, which is attached on top of the AP, in

Fig. 11(b). A clear trade-off can be observed between T_{skin} and $T_{AP,junc}$ for various types of materials. This observation complies with results reported by a group of researchers at Texas Instrument Inc. [6]. The optimal thermal path design should touch the AP junction temperature constraint and skin temperature constraint at the same time. 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 the AP and high leakage power at high temperatures) may prevent the usage of low thermal conductivity materials.

VIII. Final Remarks

A. Conclusion

We presented Therminator 2, a component-level compact-thermal-modeling-based thermal simulator targeting small form-factor devices in this work. Therminator 2 is an early-stage, full-device thermal analyzer that quickly produces accurate steady- and transient-state temperature maps of all components (ICs, boards, screens, cases, etc.) in a smartphone, from the application processor to the skin of the device, with a fast runtime. Using advanced numerical optimizations, Therminator 2 can perform steady-state simulations 1.6 times faster than the prior art technique and is capable of performing transient-state simulations in real-time and 1.25 times faster than the prior art method. It provides great flexibility in handling different user-specified design specifications and use cases. We validated temperature results produced by Therminator 2 against real temperature measurements using thermocouples and simulations using a commercial computational-fluid-dynamics tool on the Qualcomm MSM8660 MDP device and Google Nexus 5. We also provided a case study on Samsung Galaxy S4 by using Therminator 2, 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.

B. Limitations and Future Work

Therminator 2 only allows cuboid components. Other shapes such as components with circular corners can only be approximated using several small cuboids. This is a cumbersome process to be done manually and can be automated.

Besides, with the speed improvements presented in this paper, one can use Therminator 2 solver to automate the thermal design of a new device. Therminator 2 sits at the heart of an optimizer which takes into account constraints such as material properties, dimension, and ratio of various components.

Last but not least, it is important to point out that Therminator 2 inherently supports passive cooling methods through the use of materials with low thermal conductivity. However, support of active cooling techniques is more involved and is planed for future work. More concretely, the following techniques are considered.

- Thermoelectric coolers: These coolers can be divided into three layers where the bottom layer produces negative power (i.e., sucks the heat) and the top layer generates power. The middle layer generates power proportional to the Joule heating. Currently, this can be defined in the `specs.xml` file in order to associate proper power consumption to each layer. Supporting an independent thermoelectric cooler component would simplify and abstract the above details. See our earlier work for more details [20].
- Forced convection cooling (or fans): A simple and less accurate model can be added as an element with negative power. A more accurate implementation requires simulating air flow. With that said, it is important to note that very few smartphones and tables are equipped with fans due to reliability, noise, and power consumption concerns.
- One- or two-phase microchannel liquid cooling and direct jet impingement: This is a new space for portable devices and previously has been extensively studied only for high-performance devices. For instance, 3D-ICE [16] presents a CTM for these coolers. A feasibility study, including the effectiveness, cost analysis, and reliability, is required to ensure use of these coolers would be beneficial for portable devices.

References

[1] M. Pedram and S. Nazarian, "Thermal modeling, analysis, and management in vlsi circuits: Principles and methods," *Proceedings of the IEEE*, vol. 94, no. 8, pp. 1487–1501, 2006.

[2] J. Srinivasan, S. V. Adve, P. Bose, and J. A. Rivers, "The case for lifetime reliability-aware microprocessors," *ACM SIGARCH Computer Architecture News*, vol. 32, no. 2, p. 276, 2004.

[3] A. R. Moritz and F. C. Henriques, "The relative importance of time and surface temperature in the causation of cutaneous burns," *The American Journal of Pathology*, vol. 23, no. 5, pp. 695–720, 1947.

[4] E. A. Arens and H. Zhang, "The skin's role in human thermoregulation and comfort," in *Thermal and Moisture Transport in Fibrous Materials*, N. Pan and P. Gibson, Eds. Woodhead Publishing, 2006.

[5] G. L. Wasner and J. A. Brock, "Determinants of thermal pain thresholds in normal subjects," *Clinical Neurophysiology*, vol. 119, no. 10, pp. 2389–2395, 2008.

[6] S. P. Gurrum, D. R. Edwards, T. Marchand-Golder, J. Akiyama, S. Yokoya, J. Drouard, and F. Dahan, "Generic thermal analysis for phone and tablet systems," in *Proceedings of the Electronic Components and Technology Conference*, 2012, pp. 1488–1492.

[7] J. Rajmond and A. Fodor, "Thermal management of embedded devices," in *Proceedings of the International Spring Seminar on Electronics Technology*, 2013, pp. 30–34.

[8] GSMArena team, "Are gaming phones worth it? Black Shark, Red Magic vs mainstream flagships," https://www.gsmarena.com/gaming_phones_throttling_performance-review-1957.php, 2019, [Online; accessed 3-26-2020].

[9] A. Ku, "Asus Transformer Pad TF300T review: Tegra 3, more affordable," http://www.tomshardware.com/reviews/transformer-pad-tf300t-tegra-3-benchmark-review,3179.html, Apr. 2012, [Online; accessed 3-26-2020].

[10] J. A. Kaplan, "New Apple iPad hits 116 degrees, consumer reports says," http://www.foxnews.com/tech/2012/03/20/ipads-not-overheating-apple-says, 2012, [Online; accessed 3-26-2020].

[11] M.-N. Sabry, "Compact thermal models for electronic systems," *IEEE Transactions on Components and Packaging Technologies*, vol. 26, no. 1, pp. 179–185, 2003.

[12] Q. Xie, M. J. Dousti, and M. Pedram, "Therminator: A thermal simulator for smartphones producing accurate chip and skin temperature maps," in *Proceedings of the International Symposium on Low Power Electronics and Design*, 2014, pp. 117–122.

[13] "Snapdragon MDP mobile development platform - legacy devices," https://developer.qualcomm.com/hardware/mdp-mobile-development-platform, [Online; accessed 3-26-2020].

[14] "Autodesk CFD software," http://www.autodesk.com/products/cfd/overview, [Online; accessed 3-26-2020].

[15] K. Skadron, M. R. Stan, K. Sankaranarayanan, W. Huang, S. Velusamy, and D. Tarjan, "Temperature-aware microarchitecture: Modeling and implementation," *ACM Transactions on Architecture and Code Optimization*, vol. 1, no. 1, pp. 94–125, 2004.

[16] A. Sridhar, A. Vincenzi, M. Ruggiero, T. Brunschwiler, and D. Atienza, "3D-ICE: fast compact transient thermal modeling for 3D ICs with inter-tier liquid cooling," in *Proceedings of the International Conference on Computer-Aided Design*, 2010, pp. 463–470.

[17] M. J. Dousti, M. Ghasemi-Gol, M. Nazemi, and M. Pedram, "ThermTap: An online power and thermal analyzer for portable devices," in *Proceedings of the International Symposium on Low Power Electronics and Design*, Jul. 2015.

[18] Y. Han, I. Koren, and C. Krishna, "Temptor: A lightweight runtime temperature monitoring tool using performance counters," in *Proceedings of the Workshop on Temperature-Aware Computer Systems*, 2006.

[19] J. Meng, K. Kawakami, and A. K. Coskun, "Optimizing energy efficiency of 3-D multicore systems with stacked dram under power and thermal constraints," in *Proceedings of the Design Automation Conference*, 2012, pp. 648–655.

[20] M. J. Dousti and M. Pedram, "Platform-dependent, leakage-aware control of the driving current of embedded thermoelectric coolers," in *Proceedings of the International Symposium on Low Power Electronics and Design*, 2013.

[21] Z. Luo, H. Cho, X. Luo, and K.-i. Cho, "System thermal analysis for mobile phone," *Applied Thermal Engineering*, vol. 28, no. 14, pp. 1889–1895, 2008.

[22] B. Egilmez, G. Memik, S. Ogrenci-Memik, and O. Ergin, "User-specific skin temperature-aware dvfs for smartphones," in *Proceedings of the Design, Automation, and Test in Europe*, 2015, pp. 1217–1220.

[23] Y.-J. Yu and C.-J. Wu, "Designing a temperature model to understand the thermal challenges of portable computing platforms," in *Proceedings of the Intersociety Conference on Thermal and Thermomechanical Phenomena in Electronic Systems*, 2018, pp. 992–999.

[24] S. Kang, H. Choi, S. Park, C. Park, J. Lee, U. Lee, and S.-J. Lee, "Fire in your hands: Understanding thermal behavior of smartphones," in *International Conference on Mobile Computing and Networking*, 2019, pp. 1–16.

[25] H. Shen, Y. Tan, J. Lu, Q. Wu, and Q. Qiu, "Achieving autonomous power management using reinforcement learning," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 18, no. 2, pp. 1–32, 2013.

[26] F. Paterna and T. Š. Rosing, "Modeling and mitigation of extra-soc thermal coupling effects and heat transfer variations in mobile devices," in *2015 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2015, pp. 831–838.

[27] L. Zhang, B. Tiwana, Z. Qian, Z. Wang, R. P. Dick, Z. M. Mao, and L. Yang, "Accurate online power estimation and automatic battery behavior based power model generation for smartphones," in *Proceedings of the Conference on Hardware/Software Codesign and System Synthesis*, 2010, pp. 105–114.

[28] S. Li, J. H. Ahn, R. D. Strong, J. B. Brockman, D. M. Tullsen, and N. P. Jouppi, "McPAT: an integrated power, area, and timing modeling framework for multicore and manycore architectures," in *Proceedings of the International Symposium on Microarchitecture*, 2009.

[29] N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti, R. Sen, K. Sewell, M. Shoaib, N. Vaish, M. D. Hill, and D. A. Wood, "The gem5 simulator," *ACM SIGARCH Computer Architecture News*, vol. 39, no. 2, pp. 1–7, 2011.

[30] F. Kreith, *The CRC handbook of thermal engineering*. Springer, 2000.

[31] Y. Cļengel, *Heat and mass transfer: A practical approach*. McGraw-Hill, 2007.

[32] W. E. Boyce and R. DiPrima, *Elementary differential equations and boundary value problems*, 11th ed. Wiley, 2017.

[33] W. Ford, *Numerical linear algebra with applications: Using MATLAB*. Academic Press, 2014.

[34] Richard C. Allen, Chris Bottcher, Phillip Bording, Pat Burns, John Conery, Thomas R. Davies, James Demmel, Chris Johnson, Lakshmi Kantha, William Martin,, Geoffrey Parks, Steve Piacsek, Dan Pryor, Tamar Schlick, M.R. Strayer, Verena M. Umar, Robert Voigt, Jerrold Wagener, Dave Zachmann, and John Ziebarth, *Computational science education project*. U.S. Department of Energy, 1996.

[35] H. Rosenbrock, "Some general implicit processes for the numerical solution of differential equations," *The Computer Journal*, vol. 5, no. 4, pp. 329–330, 1963.

[36] "cuSOLVER | NVIDIA Developer," https://developer.nvidia.com/cusolver, [Online; accessed 7-6-2020].

[37] "Intel MKL PARDISO - Parallel Direct Sparse Solver Interface," https://software.intel.com/content/www/us/en/develop/documentation/mkl-developer-reference-fortran/top/sparse-solver-routines/intel-mkl-pardiso-parallel-direct-sparse-solver-interface.html, [Online; accessed 3-26-2020].

[38] "Superlu," https://portal.nersc.gov/project/sparse/superlu/, [Online; accessed 3-26-2020].

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

[40] "Intel math kernel library (Intel MKL)," https://software.intel.com/en-us/mkl, [Online; accessed 3-26-2020].

[41] C. Lomont, "Introduction to intel advanced vector extensions," *Intel white paper*, vol. 23, 2011.

[42] "GNU time," https://www.gnu.org/software/time/, [Online; accessed 3-26-2020].

[43] "pugixml," http://pugixml.org, [Online; accessed 3-26-2020].

[44] G. Guennebaud, B. Jacob *et al.*, "Eigen v3," http://eigen.tuxfamily.org, 2010, [Online; accessed 3-26-2020].

[45] K. Anhert and M. Mulansky, "Odeint — solving ordinary differential equations in c++," in *American Institute of Physics Conference*, 2011.

[46] "StabilityTest," https://play.google.com/store/apps/details?id=com.into.stability&hl=en, [Online; accessed 1-6-2017].

[47] "Candy Crush Saga," https://play.google.com/store/apps/details?id=com.king.candycrushsaga, [Online; accessed 3-26-2020].

[48] "YouTube," https://play.google.com/store/apps/details?id=com.google.android.youtube, [Online; accessed 3-26-2020].

[49] "Trepn Profiler," https://developer.qualcomm.com/forums/software/trepn-power-profiler, [Online; accessed 3-26-2020].

[50] P. Mochel, "The sysfs filesystem," in *Proceedings of the Linux Symposium*, 2005.

[51] "Measurement Computing's USB-2408 series," https://www.mccdaq.com/usb-data-acquisition/USB-2408-Series.aspx, [Online; accessed 3-26-2020].

[52] X. Chen, Y. Chen, Z. Ma, and F. C. Fernandes, "How is energy consumed in smartphone display applications?" in *Proceedings of the Workshop on Mobile Computing Systems and Applications*, 2013, pp. 1–6.