Modeling and Mitigation of Extra-SoC Thermal Coupling Effects and Heat Transfer Variations in Mobile Devices

Francesco Paterna and Tajana Šimunić Rosing
Department of Computer Science and Engineering
University of California, San Diego
La Jolla, California, USA
fpaterna@ucsd.edu tajana@ucsd.edu

***Abstract*— In smartphones and tablets, a number of components, such as display and communication subsystem, dissipate a significant amount of heat. This influences the SoC thermal envelope, because of the absence of a fan. Thus the thermal conditions of the SoC cannot simply be modeled as only a function of the SoC component power. In addition, contact surfaces and phone orientation variations (e.g., phone inside a pocket, phone held in a hand, and phone laying on a desk) change the heat transfer coefficients of the device over time and influence the SoC temperature. In this work, we analyze the thermal behavior of a commercial mobile device in varying user interaction profiles and under different environmental conditions. Next, we propose a system-variation aware thermal modeling strategy that only uses available power and thermal sensors. Lastly, we devise a novel ambient-aware proactive thermal management algorithm. Our approach is able to meet the given thermal constraints while providing stable performance and comparable power consumption with respect to existing techniques. In contrast, the state-of-the-art approaches, which do not consider ambient condition variations, violate the thermal constraints and lead up to 2.6x higher performance variations.**

Keywords*— *Handheld devices; dynamic thermal modeling and management

I. Introduction

While mobile SoC performance has become comparable with desktop and laptop processors for both graphic and computational applications, the increase in the SoC's power density has raised new challenges. The usability of a handheld device is limited by the battery charge duration and the device temperature. While research to date has presented multiple solutions to optimize the energy consumed by device resources such as SoC [18][5], display [15], and transceiver [12][7], the state-of-the-art on thermal modeling has mainly focused on static thermal analysis [29], which is useful at the design phase but cannot be effectively leveraged at run time.

When a handheld device extensively uses the most power hungry resources such as an SoC with GPUs and CPUs, transceiver which integrates 3/4G, WiFi, and display, the temperature of the cover increases as well as the chips' temperatures. Human skin can tolerate up to 45°C for plastic and 41°C for aluminum covers [4]. Today, many thermal sensors are embedded in phones. Sensors on the phone cover are also available. However to proactively control the thermal profile, and ensure reliability [26], a valid thermal model of the entire phone is needed. Effective approaches for proactive management of server size SoCs use linear-time invariant (LTI) system representations of the multicore processor's thermal behavior to leverage control theory techniques for proactive task migration and/or voltage-frequency scaling [3][31].

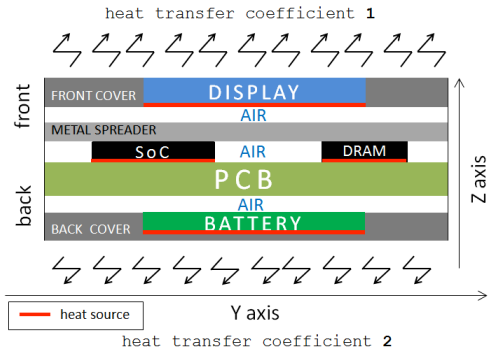


Fig. 1. Stack representation of a phone (not to scale)

Identifying the LTI equivalent thermal model for a mobile SoC has some difficulties. First, because of the absence of a fan, the various heat sources inside the device are directly thermally coupled. Second, the heat is dissipated via the natural convection [13] but the heat transfer coefficients change because of the device's orientation (i.e. vertical, horizontal) and contact surfaces (i.e., hand, pocket, blanket, and desk). Moreover, the heat flows through many different paths; the major ones are through the front and back covers. Each path has independent transfer coefficients. In conclusion, a smartphone can be described as a stack of layers built from different materials and containing several heat sources that are thermally coupled depending on the heat transfer coefficients of the device's front and back. An illustration of this is shown in Fig. 1.

978-1-4673-8388-2/15/$31.00 ©2015 IEEE

The contribution of this paper is three-fold. We first evaluate the effects of ambient variations (i.e., contact surface and orientation) on the thermal profile by measurements on our target device. We change ambient conditions and automatically replay a sequence of user-interactions over a set of mobile applications. We fix both CPU and GPU frequencies and set the "airplane mode". We observe that over a period of 10min, when the phone is lying on a desk, the SoC temperature is 1°C cooler than when it is held by a hand. If the phone is oriented vertically instead, the temperature is lower by as much as 2.5°C. Newer mobile SoCs have a larger number of CPUs and accelerators, which lead to even larger power density variations and thermal differences. Second, we propose a blind-box strategy to model the thermal behavior of the device by only accessing power and thermal sensors [16][14], as that is typically the only information that the operating system (OS) vendors have access to. The average error of our modeling approach is only 0.3°C. Last, we devise an ambient-aware thermal management technique, which manages both CPU and GPU frequency, by implementing proactive control to keep the temperature of the phone below a predefined threshold while adapting to user and environmental changes. We demonstrate through experimental results that our strategy is tolerant to ambient variations and can meet the thermal constraints. The proposed technique maintains the performance stable and provide power consumptions comparable to the traditional thermal management approaches. The state-of-the-art approaches lead up to 2.6x higher performance variations to keep the SoC in desired thermal conditions.

The remaining of this paper is organized as follows. Section II reviews the related work. Section III illustrates the analysis of the ambient variations on the target phone's thermal profile. Sections IV,V, and VI illustrate the thermal, power, and performance modeling strategy. Section VII presents the proposed management technique. Section VIII summarizes the experimental results, while Section IX concludes the paper.

II. Related Work

A number of publications have focused on developing SoC thermal models, but only a few studies to date attempted to go beyond. In [9], the thermal properties of typical smartphone components, such as printed circuit board (PCB), ICs, battery, display, and cover, have been studied separately after dissembling the device. The work in [29] discusses how the air gap between PCB and back cover impact the device's skin and the SoC temperature. The authors also analyze the impact of the materials on the leakage power. A steady-state simulator is proposed as a tool for selecting materials and dimensions of the phone. In [30], the thermal coupling between battery and SoC is studied. An RC equivalent thermal circuit is proposed and used to analyze the impact of the coupling and lead the decisions of a thermal management on selecting the best V/f setting of the CPU in order to keep the SoC temperature below a threshold. In [17], the impact of ambient temperature variations and WiFi activity on the SoC temperature are evaluated. A thermal management algorithm leverages these measurements to select the most profitable CPU frequency. The proposed model and management techniques are specific for the phone and the benchmark used. In [27], the authors use

Fig. 2. Snapdragon™ MSM 8660 by Qualcomm®.

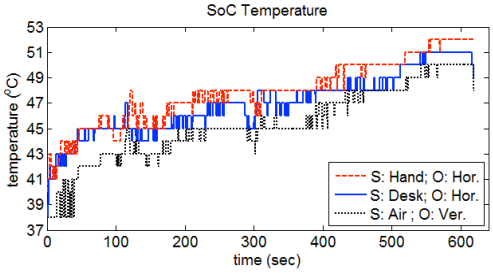


Fig. 3. Impact of contact surface (S) & orientation (O) on the thermal profile.

phase change material in mobile devices to store dissipated heat and therefore reduce the temperature. In [24], the authors first identify thermal and power models for a heterogeneous platform and then they propose a thermal and power management technique. This work does not consider ambient variation changes.

Thermal management techniques have primarily focused on multicore processors in many application domains. The most recent implementations proactively change core frequency to avoid thermal violations [3][6][23][31][10][11] while meeting performance constraints. This paper overcomes a number of major limitations of the state-of-the-art's solutions.

- RC models require physical information of the device that are often not available to OS and system vendors at the level of necessary detail. Our first contribution is to get the thermal model of commercial phones by only accessing power and thermal sensors, a much simpler but still accurate approach.
- Previous papers have focused on the internal thermal characteristics of mobile devices, disregarding the fact that the heat internally generated also depends on the device's orientation and contact surfaces. Our model and management technique is aware of this behavior.
- State-of-the-art solutions only control the CPU frequency even if the GPU currently consumes more power in mobile SoCs. Instead, our management technique controls V/f setting of the both resources separately, thus delivering a more efficient and higher performing solution.

III. METHODOLOGY

A. Target Device

The smartphone used in this paper is the Snapdragon™ Development Platform by Qualcomm® (see Fig. 2) [20]. This device has a MSM8660 SoC with two ARM15 [2] CPUs and Adreno™ GPU [19]. The OS is Android 2.3 Gingerbread. We use the Trepn profiler [21] to collect power data from the CPUs, the GPU, the DRAM, the display, and the battery. The sample time is set to 0.1sec. By accessing the virtual system sysf through the Android Debug Bridge (ADB), we obtain the SoC temperature. We can also set the two CPU and the GPU frequency independently from user-space from /sys/devices/system/cpu and /sys/class/kgls, respectively. Although we use this phone to develop and verify our model, the same modeling procedure can work with any mobile device. We use record-and-replay routines [8] to replay user-interactions on specific mobile applications with the goal of evaluating the impact of ambient variations on the performance, the temperature, and the power consumption of CPU and GPU at different frequency settings. During our experiments, we keep the phone in airplane mode as it was not possible to measure power and temperature of the transceiver.

B. Impact of Ambient Conditions

We analyze the impact of different contact surfaces and phone's orientation on the thermal profile of the device. We set the CPU frequency to 1080MHz and GPU to 200MHz, and we record a sequence of user interactions with a set of apps such as ONE Browser, Photo Gallery, VideoPlayer, and InkPad (text editor), which are available application on Amazon App Store [1]. For the browser, we use a set of locally stored websites. Interactions consist of actions such as zooming in and out, scrolling up and down, typing, and others. The duration of the recorded session is ~10min. We evaluate three different scenarios characterized by orientation (O) and contact surface (S). First, the phone is in a horizontal position lying on a desk. Then, the phone is horizontal but held by a hand. Last, the phone is standing vertically with no contact to display or back cover. Before each run, we left the phone charging and idle for 10 minutes to ensure all experiments start at the same initial temperature, which is 38°C.

We measured the SoC temperature and plotted it in Fig. 3. The dotted black curve is phone standing vertically with no contact surface (i.e., S=air), the blue line represents when the phone is lying on a desk in a horizontal position, while the dashed red is for the case when the phone is held by a hand in a horizontal position. Because our test cases are replayable and frequencies are fixed, the differences in temperature are mainly due to different ambient conditions. When the phone is on the desk surface and horizontally oriented or when it is vertically surrounded by air, it is cooler than when it is held by a hand by 1°C & 2.5°C respectively. Newer mobile SoCs have a larger number of CPUs and accelerators that lead to even larger thermal differences. Since such ambient condition variations are common in practice, it is important to account for the effect they cause on the SoC temperature. Thus, in the next sections we describe a new thermal modeling strategy combined with a management technique for ambient-aware thermal control with the goal of meeting thermal constraints while reducing the impact on performance.

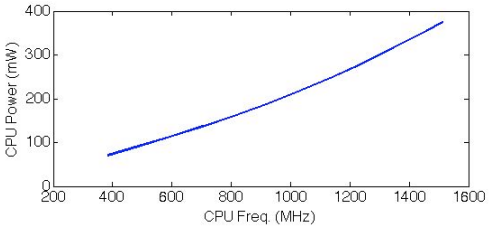


Fig. 4. CPU power versus frequency model.

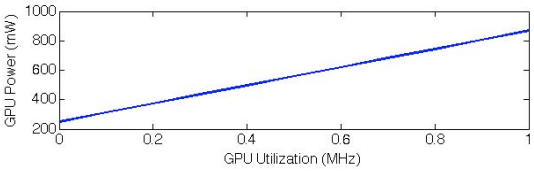


Fig. 5. GPU power versus utilization model.

IV. THERMAL MODEL

Our goal is to identify a thermal modeling strategy that can be easily used by OS vendors for any mobile device. Thus, we assume that we have no access about the detailed information describing topological and physical parameters of the phone (e.g. we do not know material characteristics and layers of the phone's PCB board). We also assume that our system can access coarse grained power and thermal sensors of the phone's key heat sources. Indeed, such information are available in today's devices [14][16].

Let the number of the heat sources be n. $\vec{T}_k \in \Re^n$ and $\vec{P}_k \in \Re^n$ represent the vectors of the temperature and the power at instant k. $A, B \in \Re^{n \times n}$ are defined as the state and the input matrices. The order of the model is equal to the number of the heat sources (i.e., n), which is usually small. In this paper we use n=6. The output of the model is the predicted temperature at time k+1, given input of currently measured temperatures and power consumption at time k. The equivalent state-space model of the smartphone's thermal behavior is given in Equation (4.1). It is represented at 0.1sec resolution as we get power and thermal measurements at the same granularity.

$$\vec{T}_{k+1} = A \cdot \vec{T}_k + B \cdot \vec{P}_k \tag{4.1}$$

Deriving the model (i.e. matrices A & B) of Equation (4.1) by only accessing power and temperature is a blind identification problem. To solve this problem, the numerical algorithm for subspace system identification, N4SID, offers several advantages. It is not iterative and the initial state does not have to be specified. N4SID has two steps: (i) a state sequence of the system is recognized by projecting an input and an output sequence; (ii) the matrices of the state-space representation of the system are recognized by using a least squares approach. More details are provided in [28].

V. POWER MODEL

To control the power, and thus the temperature, we base on V/f scaling of the SoC's units. To act a proactive management control we need power and performance models. We discuss the CPU and the GPU power model in Sections V.A and V.B while we illustrate the performance model in Section VI.

A. CPU Power Model

For a CPU, the power is the sum of two contributions: dynamic and leakage. The dynamic power can be modeled through Equation (5.1) where α an C are the activity factor and the switching capacitance.

$$P_{dyn-CPU} = \alpha C V_{dd-CPU}^2 f_{CPU} \tag{5.1}$$

The leakage power can be modeled through Equation (5.2) where T represent the temperature; the coefficient b_1 accounts technology dependent constants, channel length, and width; the coefficient b_2 accounts the Boltzmann constant, the electron charge, and the threshold voltage; and I_{gate} is the gate leakage current that can be assumed constant.

$$P_{lkg-CPU} = V_{dd-CPU}\left(b_1 T^2 \exp\left(\frac{b_2}{T}\right) + I_{gate}\right) \tag{5.2}$$

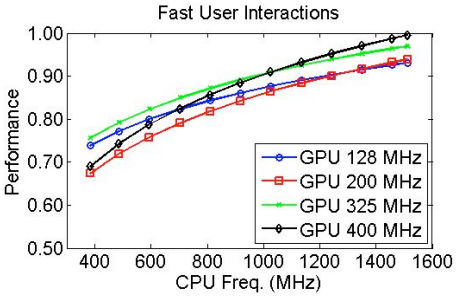
As the frequency f linearly increase with respect to the V_{dd}, the dynamic power can be approximated as a cubic function of the frequency while the leakage power can be approximated as linear function of the frequency. To identify the CPU power model we have executed several registrations of user interactions such as zooming and scrolling when using Gallery, ONE Browser, Adobe PDF Reader, which are mobile applications available on Amazon App Store [1]. Each run had fixed CPU and GPU frequency. We have evaluated all the possible frequency configurations. We have measured the power consumption for each run. We have seen that both thermal and GPU frequency variations do not significantly influence the CPU power that we have thus modeled as function of the frequency as illustrate Equation (5.3) and plotted in Fig. 4. We have identified the coefficients a_1 and a_2 by executing the nonlinear least square fitting over the collected data. The approximation error of the power model over the collected data is below 4%.

$$P_{CPU}(f_{CPU}) = a_1 f_{CPU}^3 + a_2 f_{CPU} \tag{5.3}$$

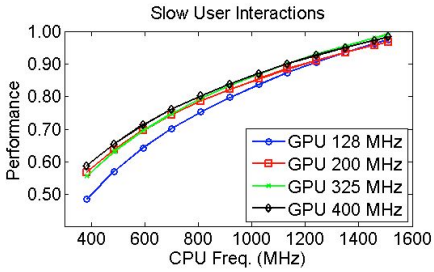
B. GPU Power Model

We have seen the GPU power seems mostly correlated to the GPU utilization U_{GPU} because in our system the voltage of the GPU remains fixed over frequency changes. Therefore, we describe the GPU power P_{GPU} through Equation (5.4). We have identified the coefficients b_1 and b_2 by executing the linear least square fitting over the collected data. Fig. 5 plots the GPU power model versus the GPU utilization. The approximation error of the power model over the collected data is below 4%.

$$P_{GPU}(u_{GPU}) = b_1 u_{GPU} + b_2 \tag{5.4}$$



a) Fast interactions (i.e., scrolling through pictures)



b) Slow interactions (i.e., reading news)

Fig. 6 Performance versus CPU frequency at different GPU frequencies

VI. PERFORMANCE MODEL

The Quality of Experience (QoE) of the device can be evaluated by verifying that the system responds sufficiently fast to the user interactions. Some applications, such as video playback, do not have frequent user interactions. The frame rate that guarantees satisfactory execution of the video is usually relatively low, such as 25 frames per second (fps). Our target device can meet 25fps at fairly low CPU and GPU frequency as long as there are not too many other active processes. In contrast, when scrolling or zooming, the phone tries to execute the workload at 60fps, maximum observed frame rate. Therefore, for the purposes of this work, we quantify performance as a normalized frame rate.

Performance is a function of both CPU and GPU frequencies. Memory traffic impacts the computational efficiency of the SoC. We measure the frame rate for a set of replayable runs, each one executed at different CPU and GPU frequencies. We divide the runs into two categories: fast, when the user is scrolling and zooming quickly (e.g. looking for a picture or a detail in some pictures) and slow, when the user is scrolling or zooming at lower speeds (i.e., the user reads contents of a webpage). We examine all the possible CPU and GPU frequency configurations. Each run lasts approximately 10 seconds. We measure the average frame rate per run. Equation (6.1), which is a function of the CPU frequency f_{CPU}, provides a sufficiently accurate performance model of our platform. The coefficient c_1 and c_2 of Equation (6.1) depend on GPU frequency. We obtain the values for these coefficients during *fast* and *slow* interactions with nonlinear least square fit over the collected data. Fig.s 6a) and 6b) plot the performance versus the CPU frequency at

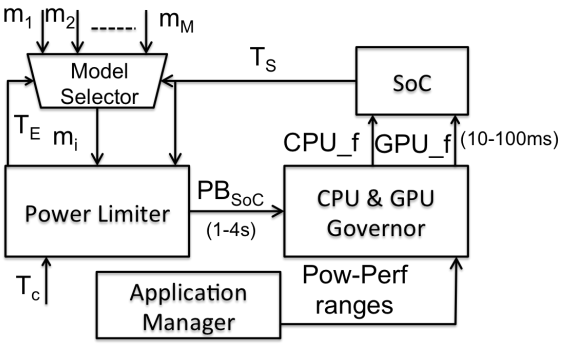


Fig. 7. APTM block diagram.

different GPU frequencies. The error of the model as compared to the collected data is less than 6%.

$$FPS(f_{CPU})\big|_{f_{GPU}} = c_1 \log(c_2 f_{CPU}) \qquad (6.1)$$

Fig.s 6a) and 6b) shows that the maximum GPU frequency does not always provide maximum performance. For instance, during fast interactions, when the CPU frequency is below 1026 MHz, better performance is provided by the GPU clocked at 325 MHz. This happens because the high GPU frequency creates more unproductive traffic to memory (i.e., polling).

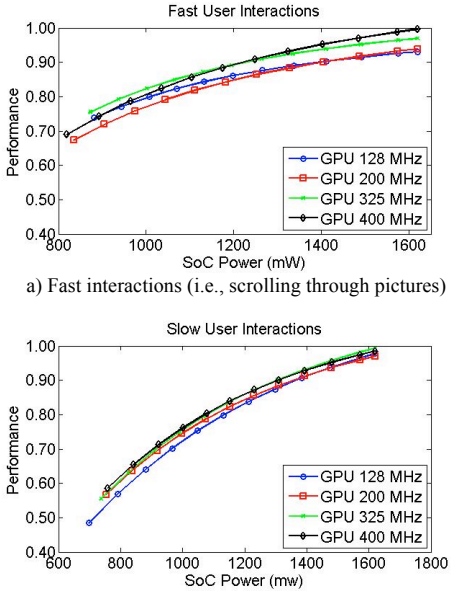
In addition, we model GPU utilization as a function of performance. GPU is fully utilized (i.e. U_{GPU}=1) when the frame rate is maximum possible one a given GPU frequency; so we model the GPU utilization as shown in Equation (6.2).

$$U_{GPU}(f_{CPU})\big|_{f_{GPU}} = FPS_{GPU}(f_{CPU})\big|_{f_{GPU}} / \max(FPS|_{GPU}) \qquad (6.2)$$

VII. Ambient-aware Proactive Thermal Management

We propose an Ambient-aware Thermal Management (APTM), which aims to meet a given thermal constraint while maximizing the performance. APTM uses a set of thermal models obtained offline using our strategy presented in Section IV. The main APTM's components are four. Power Limiter computes the SoC power budget that guarantees the meeting of the thermal constraint Tc. Power Limiter takes the thermal model from Model Selector and reads the current SoC temperature from the sensor. Governor reads the power budget and application information from Application Manager to set the CPU and the GPU frequency. Model Selector updates the thermal model with respect to ambient condition changes. APTM's block diagram is shown in Fig. 7.

Power Limiter proactively fixes an SoC power budget PB_{SoC} so that the estimated SoC temperature is less than the constraint T_c (i.e., $T[k] < T_c$) for a period τ, which ranges from 1 to 4 sec. It uses the thermal model of Equation (4.1) and the initial temperature T_s, which is read from the sensors. Starting from the maximum SoC power (i.e., the power obtained at the maximum CPU and the maximum GPU frequency), Power Limiter checks whether the estimated SoC temperature is below the threshold for every sample k. If τ =1 sec, the number of samples to check is 10. If not, the power is



a) Fast interactions (i.e., scrolling through pictures)

b) Slow interactions (i.e., reading news)

Fig. 8. Performance versus power consumption at different GPU frequencies.

decreased by 0.1W and the comparison is repeated until either the constraint is met or the power is as much as the minimum value (i.e., the power value obtained at the minimum CPU and the minimum GPU frequency). Power Limiter uses a thermal model shown in Equation (4.1), which requires data from six power sources. Three of them are related to the SoC (i.e., 2 CPUs and 1 GPU) while the others are display, battery, and DRAM. The current budget of PB_{SoC} is divided by assuming that ¼ PB_{SoC} is consumed by each CPU and ½ PB_{SoC} is consumed by the GPU. For power consumed by display, DRAM, and battery, it is assumed that those sources will consume the same power as during the last period τ. The power data can be obtained via power sensors [16].

Application Manager evaluates performance as a function of SoC power by using models of Sections V and VI. Fig.s 8a) and 8b) shows the trends of performance versus power at different GPU frequencies for fast and slow interactions. Fig. 8a) shows that for fast interactions there are three significant power ranges. Until 0.9W, GPU running at 400MHz provides better performance. From 0.9 to 1.2W GPU at 325MHz is the best, while at power values larger than 1.2W, the GPU frequency of 400MHz is again more appropriate. In contrast, when iterations are slow (see Fig. 8b)), if the power budget is not larger than 0.75W the best GPU frequency is 128MHz. From 0.75 to 1.2W the GPU frequency is set to 400MHz. If the power budget is larger than 1.2W, GPU frequency of 325MHz is the best. Application Manager stores these power ranges and depending on the current application, it provides this data to Governor. This approach is not limited to only to fast and slow interactions but it can be extended to more types of workload.

Governor selects CPU and GPU frequency at every scheduling tick. It relies on a set of power ranges provided by the Application Manager. The Governor sets the GPU frequency depending on the current power budget provided by

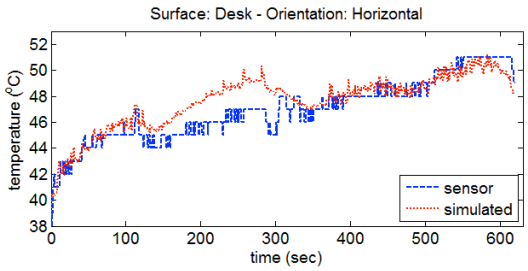


Fig. 9. SoC temperature obtained from the sensors (dashed blue curve) versus the simulated (dotted red curve)

Power Limiter and subsequently it selects the CPU frequency as described next. Starting from the highest frequency, it computes the total power by using the model of Equation (5.3) and compares it to the budget. Next, it decreases the CPU frequency until either the estimated total power is not larger than the budget or the CPU frequency is the lowest possible.

Model Selector updates the thermal model for Power Limiter. It has stored a set of precomputed thermal models generated for different ambient conditions. The models are sorted from the coolest to the hottest as a function of the modeled temperature. Model Selector reads the current temperature from the sensor Ts and compares it against the estimated one Te from Power Limiter. If the difference is smaller than 1°C no action is taken. Otherwise, if Ts > Te , the current model is replaced with the next hotter one, else the next cooler model is chosen.

VIII. RESULTS

A. Experimental Setup

We next evaluate the effectiveness of our models and management techniques presented in Sections IV and VII. For all measurements we use the phone described in Section III.A. Six different heat sources are taken into account: CPU1, CPU2, GPU, DRAM, display, and battery. Our modeling and management strategies need as many power and temperature sensors but the phone only provides one SoC thermal sensor. Therefore, we combine the measurements we obtain from the phone with a simulation infrastructure.

We create a thermal model of the whole phone by using 3D-ICE, a tool for transient thermal modeling of 3D IC structures [25]. 3D-ICE is suitable for modeling the thermal behavior of smartphones for a few reasons. First, it is possible to set the natural convection via two different paths to the ambient, top & bottom, with their respective heat transfer coefficients [W/m^2-K]. As a result, we can model a parallelepiped for which the heat internally generated is transferred to the ambient through the top and bottom face (e.g. phone's display and the back cover). We used a calibration procedure to set the two heat transfer coefficients. For each ambient condition analyzed in Section III.B, we found the two coefficients that match the simulated SoC temperature with the measured one. Second, the parallelepiped can be modeled as stack of layers composed of different materials. For each layer, dimensions, thermal conductivity [W/m-K] and volumetric heat capacity [J/m^3-K] of the material can be specified. Different material characteristics across the same layer can be specified to model a number of components mounted on a PCB and separated by air gaps.

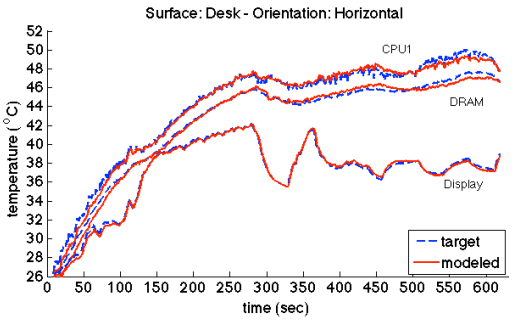


Fig. 10. Comparison between the CPU1, DRAM, and Display temperatures using the model found with N4SID (dotted red) versus the target traces (dashed blue) for the ambient condition {desk;oriz.}.

Fig. 9 shows the thermal results we obtained with phone lying on a desk. The thermal simulation with 3D-ICE (dotted red curve) is only an approximation of the real thermal behavior (dashed blue curve) since the heat is not dissipated uniformly through the front and back faces of phone. Also, even if we access the phone's thermal sensors every 100ms, the register may not been updated as often, resulting in the difference between the simulated and measured data.

B. Modeling

In this section we show that the identification approach of Section IV can find an accurate thermal model of smartphones in various ambient conditions. We store the power traces of CPU1, CPU2, GPU, DRAM, display, and battery using Trepn [19] for a set of sessions running on the phone described in Section III.A, roughly 10min long. We use the following mobile applications: ONE Browser, InkPad, Galley, and VideoPlayback. Frequency governor is set to ondemand. We provide the power traces to the phone's thermal simulator configured in three different ways related to the ambient conditions analyzed in Section III.B: {desk; horiz.}, {hand; horiz.}, and {air; vertical}.

Using N4SID, we generate three different state-space models described by Equation (4.1) (i.e., A & B matrix) for each ambient condition (i.e., contact surface and orientation). The order is set to the number of heat sources (6). The average error, when comparing both training and test sets is 0.3°C while the maximum error is below 1.5°C. For the case {desk; horiz.}, the CPU1, DRAM, and Display target temperatures (blue dashed curves) versus the ones estimated with the model found by N4SID (red dotted curves) are shown in Fig. 10.

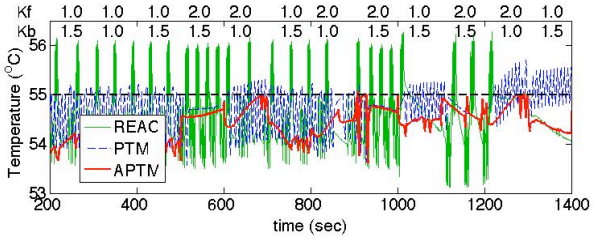


Fig. 11a). SoC temperature comparison.

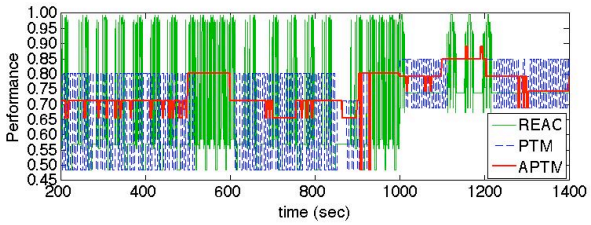


Fig. 11b). Performance comparison.

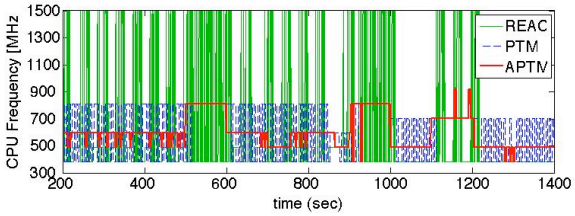


Fig. 11c). CPU frequency comparison.

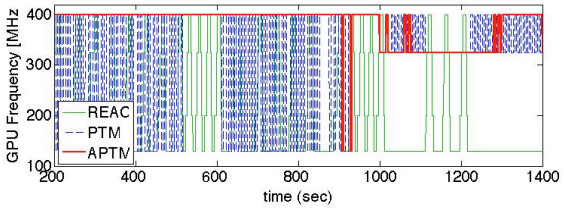


Fig. 11d). GPU frequency comparison.

We also evaluate the time required by N4SID on a computer with an Intel Quad Core Q8300 running at 2.5GHz. We changed the number of inputs and outputs from 5 to 7 and the number of samples from 1000 to 2000. It took only 9sec on average to compute. This procedure is always executed offline (once for each ambient condition) while the resulting models are used by the management technique online as explained in Section VII and evaluated in Section VIII.C.

C. *Management*

In this experiment, we demonstrate the effectiveness of the proposed Ambient-aware Proactive Thermal Management (APTM) technique presented in Section VII. We a simulation infrastructure composed of thermal, power, and performance models illustrated in Sections IV, V, and VI.

We gathered the power traces from our target device using Trepn [21]. We execute multiple runs on our target phone. During each run, we scroll and zoom on the phone to simulate user interaction. We used Gallery, ONE Browser (for which we navigated through locally stored webpages), and Acrobat PDF Reader, which are mobile applications available on Amazon App Store [1].

We used several thermal models, each one related to a different ambient condition. We considered a set of nine heat transfer coefficient configurations $\{K_f, K_b\}$, in which K_f refers to the heat transfer coefficient on the front of the device (see Fig. 1) while K_b refers to the back (see Fig. 1). In each simulation, we randomly change the ambient conditions (i.e., the heat transfer coefficients) every 100sec. The normalized values are reported on the top x-axis of Fig. 11a.

We compare the proposed APTM against the Reactive Thermal Management (REAC) technique, akin to [22], & ambient-agnostic Proactive Thermal Management (PTM) technique, akin to [24].

REAC [22] activates every 0.1sec and operates CPU and GPU frequency scaling. If the temperature read from the sensor T_S is larger than the thermal constraint T_C, the CPU frequency is reduced by one step unless the CPU frequency is already set at the lowest level. In the latter case, the GPU frequency is decreased by one step and the CPU frequency is set to the maximum value. If $T_S < T_C - \delta$, where δ is set to 1^oC, the complementary approach is used to increase the performance. PTM executes the APTM scheme but it does not update the thermal model (i.e. Model Selector) as is customary in the state-of-the-art's model predictive control techniques [24].

TABLE I. Performance variations. Standard deviation over average.

	REAC [22]	PTM [24]	APTM
σ / μ	2.60	2.25	1.00

The three SoC thermal envelopes are shown in Fig. 11a while the performances are plotted in Fig. 11b. The CPU and GPU frequency are plotted in Fig. 11c and 11d, respectively. In all cases the initial temperature is 38^oC. The curves are plotted after the warm-up phase that lasted approximately 200sec. The proposed APTM is the only policy able to meet the thermal constraint, set to 55^oC. When the heat conduction capability is high (i.e., $\{K_f=2.0, K_b=1.5\}$ at 500, 900, and 1100 sec), APTM recognizes this and increases performance. PTM slightly violates the thermal constraint and generates wider performance variations. PTM uses as initial temperature the one read from the sensor so even if its thermal model is not accurate, it is able to prevent large thermal violations but it uses wider CPU and GPU frequency changes. REAC continuously violates the thermal constraints, has large CPU and GPU frequency variations leading to erratic performance.

As Fig. 11b shows, APTM has less performance variations than either state of the art technique. APTM's minimum performance is always better than the minimum performance of PTM and REAC. To further quantify the stability of delivered performance, we calculate for each technique the ratio between the standard deviation of the performance (σ) and the average of the performance (μ), and then normalized results to our technique, APTM, as it has by far the lowest

performance variations. The results are shown in Table I. In comparison to APTM, PTM and REAC, have 2.25x and 2.6x higher performance variations, a big issue when it comes to user interaction. The power consumed by the APTM and PTM is comparable and equal to 0.6W on average. REAC consumes as much as 1W on average. In summary, the results demonstrate the importance of using a model which is aware of ambient condition variations and that the proposed strategy is effective in controlling the thermal profile of the SoC while guaranteeing stable performance.

IX. Conclusion

In this work, we address the problem of heat transfer variations and thermal coupling between components in smartphones. We propose a proactive control technique supported by a novel modeling strategy to keep the device below a predefined thermal threshold while minimizing performance variations. Our results show that the existing techniques, which are not aware of ambient condition variations, lead to thermal violations and 2.6x higher performance variations, while our proposed method meets the thermal constraints and keeps performance stable.

Acknowledgment

The research has been supported by NSF SHF: Small: Cooling, energy and performance management in computing systems.

References

[1] Amazon App Store. Amazon.com April 2015 [online]

[2] ARM Ltd. Cortex A15. July 2014.

[3] Bartolini A., Cacciari M., Tilli A., and Benini, L., "A distributed and self-calibrating model-predictive controller for energy and thermal management of high-performance multicores," In *Proc. of the Int. Conf. on Design, Automation & Test in Europe*, pages.1-6, IEEE, 2011.

[4] Berhe, M.K., "Ergonomic Temperature Limits for Handheld Electronic Devices", In Proceedings of *InterPACK*, ASME, 2007.

[5] Chen X., Chen Y., Dong M., and Zhang C.. "Demystifying Energy Usage in Smartphones." In *Proc. of the Design Automation Conference*, pages 1-6, ACM, 2014.

[6] Coskun, A.K.; Rosing, T.S.; Gross, K.C., "Utilizing Predictors for Efficient Thermal Management in Multiprocessor SoCs," *Computer-Aided Design of Integrated Circuits and Systems*, IEEE Transactions on , vol.28, no.10, pages 1503-1516, Oct. 2009.

[7] Ding N., Wagner D., Chen X., Pathak A., Hu Y C., and Rice A."Characterizing and modeling the impact of wireless signal strength on smartphone battery drain", In *Proc. of the Int. Conf. on Measurement and Modeling of Computer Systems*, pages 29-40, ACM, 2013.

[8] Gomez L., Neamtiu I., Azim T., and Millstein T. "RERAN: timing- and touch-sensitive record and replay for Android", In *Proc. of the Int. Conf. on Software Engineering,* pages 72-81. IEEE, 2013.

[9] Gurrum S., Edwards D., Marchand-Golder T., Akiyama J., Yokoya S., Drouard J., and Dahan F. Generic thermal analysis for phone and tablet systems. In *Proc. of the Int. Conf. on Electronic Components and Technology*, pages 1488–1492. IEEE, 2012.

[10] Hanumaiah V. and Vrudhula, S. "Energy-Efficient Operation of Multicore Processors by DVFS, Task Migration, and Active Cooling," Computers, IEEE Transactions on , vol.63, no.2, pp.349,360, Feb. 2014.

[11] Hanumaiah V., Desai D., Gaudette B., Wu C., and Vrudhula S. "STEAM: A Smart Temperature and Energy Aware Multicore Controller," ACM Trans. Embed. Comput. Syst. 13, 5s, Article 151, Oct. 2014.

[12] Huang J., Qian F., Gerber A., Mao M. Z., Sen S., and Spatscheck O. "A close examination of performance and power characteristics of 4G LTE networks". In *Proc. of the Int. Conf. on Mobile systems, Applications, and Services*, pages 225-238, ACM, 2012.

[13] Incropera FP and DeWitt DP, "Foundamentals of Heat and Mass Trasnfer" John Wiley and Sons, 1996.

[14] Intel® 64 and IA-32 Architectures Software Developer's Manual - Volume 3B. Intel Corporation, Jun. 2009.

[15] Kim D., Jung N., and Cha H. Content-centric Display Energy Management for Mobile Devices. In *Proc. of Design Automation Conference*, pages 1-6, ACM, 2014.

[16] Maxim Integrated™ 78M6631 3-Phase Power Measurement and Monitoring SoC data sheet, Arp 2012.

[17] Paterna F., Zanotelli J. and Rosing T. "Ambient variation-tolerant and inter components aware thermal management for mobile system on chips." In *Proc. of the Int. Conf. on Design, Automation and Test in Europe*, pages 1-6, IEEE, 2014.

[18] Pathania A., Jiao Q., Prakash A., and Mitra T. "Integrated CPU-GPU Power Management for 3D Mobile Games." In *Proc. of the Design Automation Conference*, pages 1-6, ACM, 2014.

[19] Qualcomm developer network. Adreno. April 2015 developer.qualcomm.com/discover/chipsets-and-modems/adreno-gpu [online].

[20] Qualcomm developer network. Snapdragon™ S4 Plus MSM8960 MDP. April 2015 developer.qualcomm.com/mobile-development/development-devices/snapdragon-s4-msm8960-mdps [online].

[21] Qualcomm developer network. Trepn. April 2015 developer.qualcomm.com/mobile-development/performance-tools/trepn-profiler [online]

[22] Rodero I., Lee E., Pompili D., Parashar M., Gamell M., and Figueiredo, R., "Towards energy-efficient reactive thermal management in instrumented datacenters," In *Proc. of the Int. Conf. on Grid Computing,* pages 321-328. IEEE/ACM, 2010.

[23] Sharifi, S.; R. Ayoub, and T. Rosing. Tempomp: Integrated prediction and management of temperature in heterogeneous mpsocs. In *Proc. of the Int. Conf. on Design, Automation Test in Europe Conference Exhibition*, pages 593–598. IEEE, 2012.

[24] Singla G., Kaur G., Unver A, and Ogras U. "Predictive Dynamic Thermal and Power Management for Heterogenous Mobile Platforms," In *Proc. of the Int. Conf. on Design, Automation and Test in Europe*, pages 1-6, IEEE, 2015.

[25] Sridhar A., Vincenzi A., Ruggiero M., Brunschwiler T., and Atienza D. 3d-ice: Fast compact transient thermal modeling for 3d ics with intertier liquid cooling. In *Proc. of the Int. Conf. on Computer-Aided Design*, pages 463–470. IEEE, 2010.

[26] Srinivasan J, Adve S., Bose P., and Rivers J. "Lifetime Reliability: Toward an Architectural Solution" *Micro*, 25(3):70–80, IEEE, 2005.[24]

[27] Tan, F.L.; Fok, S. C., "Thermal Management of Mobile Phone using Phase Change Material,", In *Proc. of Electronics Packaging Technology Conference*, pages 836-842, IEEE, 2007.

[28] Van Overschee, P. and De Moor, B. "N4SID: Subspace Algorithms for the identification of Combined Deterministic-Stochastic Systems" In *Automatica* 30(1), pages 75-93. Pergamon Press 1994.

[29] Xie Q., Dousti M.J., and Pedram M. "Therminator: a thermal simulator for smaprthones producing accurate chip and sking temperature maps", In *Proc. of the Int. Symp. on Low Power Electronics and Design*, pages 117-122, ACM, 2014.

[30] Xie Q., Kim J., Wang Y., Shin D., Chang N., and Pedram M. "Dynamic thermal management in mobile devices considering the thermal coupling between battery and application processor", In *Proc. of the Int. Conf. on Computer-Aided Design*, pages 242-247, IEEE, 2013.

[31] Zanini F., Atienza D., Benini L., and De Micheli G. "Multicore thermal management with model predictive control," In *Proc. of the Int. Conf. on Circuit Theory and Design*, pages 23-27, IEEE, 2009.