ICCAD : G : Pushing QoS-Awareness into Thermal Management for Sustainable User Experience in Mobile Devices

Onur Sahin
Boston University, Boston, MA, USA
sahin@bu.edu

1. PROBLEM AND MOTIVATION

Aggressive CPU designs (i.e., out-of-order and speculative multi-issue pipelines [22]) and integrated GPUs have significantly increased the peak power density in today's mobile system-on-chips (SoCs). For instance, the Snapdragon 801 SoC (found in Samsung S5) can dissipate up to 8W despite the 3.5W thermal design power (TDP) budget [13][1]. As a result of excessive power densities, maximum chip temperatures can quickly elevate to critical levels while running computationally demanding applications (i.e., browsing, gaming etc.)[30, 32]. The problem becomes even more apparent and crucial for mobile devices (i.e., smartphone, tablet etc.) due to inherent limitations in cooling capabilities imposed by small form-factor requirements. Throttling the mobile processor is the current practice for reducing temperature. Thermal throttling, however, can incur severe degradations in user quality-of-service (QoS)[2] due to CPU slowdown.

Traditionally, thermal management research focused on maximizing performance under thermal restrictions [9, 11, 30], assuming that such approach will always maximize the user experience. The unique observation that motivates our research falsifies this assumption: *greedily exhausting the thermal headroom improves short-term user-experience, but at the expense of significantly less QoS over extended durations*. Our recent research [27, 28] revealed that rapidly elevated on-chip and external component (i.e., battery, display etc.) temperatures can lead to increasingly aggressive thermal throttling over time and cause as much as 50% QoS loss. Mobile users, however, expect consistent and acceptable QoS while running applications for minutes or longer. In fact, users have also already reported significant performance loss and dissatisfaction during extended durations of device use [7, 15]. Providing users with *longer durations of sustainable QoS*, thus, requires thermal management policies to deviate from existing greedy approach and adopt thermally-efficient strategies that can make more conservative usage of thermal headroom. This motivation forms the basis of our novel *QoS-centric thermal management* approach. The main objective of this work is to mitigate throttling-induced QoS degradations by slowing down the heating while delivering 'just enough' [28] (as opposed to the best) QoS for meeting given user performance requirements.

[1]TDP refers to the maximum heat generated by the SoC that the system can dissipate via natural convection or cooling.

[2]QoS refers to a metric used to quantify the performance experienced by the user (e.g., frames-per-second (FPS) or response latency). While QoS could be application specific, our techniques are not specific to metric selection and can be applied broadly.

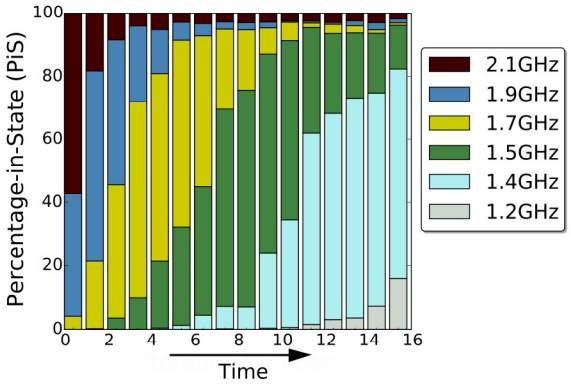

Figure 1: Frequency residencies over time on a MDP8974 smartphone during continuous use. Performance impact of thermal throttling increases over time as CPU is forced to use lower frequencies to meet the thermal constraints.

Demonstrating the problem on real-life systems. Consider the case in Figure 1 which shows the frequency residencies over time on a Snapdragon MDP8974 smartphone platform [26] while running a common FFT kernel [25] repeatedly. The default Linux CPU power manager scales the frequency to the highest level (2.1GHZ) to boost performance. Over time, however, there is a clear shift towards utilizing lower frequencies due to more aggressive thermal throttling which, significantly reduces performance. Figure 2 shows the QoS degradation over time for various real-life applications (involving gaming, media streaming) running on a Odroid-XU3 [5] mobile platform that integrates an emerging heterogeneous multi-core CPU architecture (i.e., big.LITTLE[3]). Over an 8 minute continuous use, throttling incurs significant QoS loss over time for all applications, reaching up to 50% degradations for the Aquarium application. While heterogeneous CPU designs can significantly improve energy-efficiency [24, 29, 35], clearly, thermal challenges continue to endanger user-experience over long durations of device use.

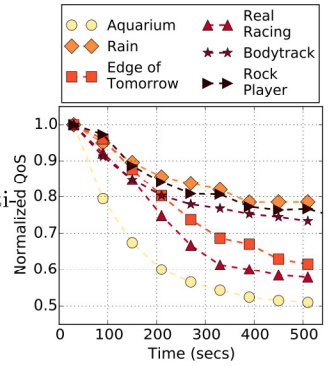

Figure 2: QoS degradation over time on Odroid-XU3 platform[5].

2. BACKGROUND AND RELATED WORK

Temperature management techniques have been widely studied in the literature and applied in commercial processing platforms. Control theoretic [9] and predictive [30] techniques are among the most popular techniques for achieving maximum performance while providing smooth control over the maximum chip temperature. While prior work focuses on achieving the best performance out of available thermal headroom (i.e., difference in current and maximum allowable skin and chip temperatures), we demonstrate the performance drawbacks of such approaches over extended durations of device activity.

A new direction in thermal modeling and management of mobile devices looks beyond the silicon and considers skin level temperatures [12] as well as the thermal interactions of the processor with surrounding components and environment (i.e., battery, air temperature etc.). Xie et al. [32] find that elevated battery temperatures can further increase the chip temperature. While my initial studies show similar findings [27], our research focuses on understanding the performance impact of such thermal interactions. Prior work on QoS management only focuses on energy minimization [16, 17, 34]. My work utilizes QoS awareness and management to improve thermal efficiency and to provide the mobile device users with desirable QoS levels while they are interacting with their devices for minutes or longer durations.

3. APPROACH AND UNIQUENESS

Our work is the first in the field to demonstrate the performance drawbacks of existing thermal management techniques over extended durations of activity. As opposed to greedily maximizing user experience under thermal constraints, we propose to integrate the user/application QoS requirements into thermal management for utilizing the thermal headroom more efficiently. This research seeks to enable *thermally-efficient QoS management* to provide the mobile system users with longer periods of acceptable performance. We design a practical QoS management approach that takes the target QoS requirements as the primary constraint and applies dynamic frequency management (DVFS) and thread scheduling control knobs in runtime to deliver desired performance. We develop several novel insights to improve the thermal-efficiency of QoS management.

(1) QoS-Temperature Tradeoff: We demonstrate that users or the system-level policies can proactively tradeoff QoS for an application that suffer from thermal throttling to gain extra thermal headroom that can be utilized for longer durations. The idea of trading off the application QoS is not completely new as such insight has been widely used for energy savings [16, 17, 34]. Trading off the QoS for thermal headroom, however, is a novel insight brought by our work to address the aforementioned QoS unsustainability problem in mobile devices.

Figure 3 illustrates this intuition on a real hardware. We show the QoS (frames-per-second (FPS)) and temperature traces for a gaming application we ran under two different CPU frequency management policies [1] in Android. While the default *Interactive* governor provides the highest QoS initially, it quickly (in 40 seconds) exhausts the processor thermal headroom and exhibits significant QoS degradation over time due to thermal throttling. *Powersave* governor, which prioritizes lower frequencies, is able the sustain the

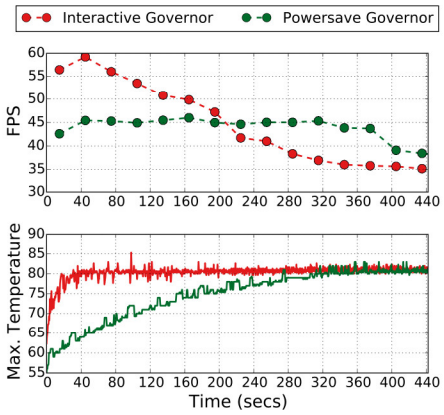

Figure 3: A case for QoS-temperature tradeoff for achieving longer sustainable QoS for a gaming application (*Real Racing*) running on Odroid-XU3 [5] mobile platform.

QoS above 45FPS for almost twice as long. This is achieved by mitigating the thermal throttling further to 380 seconds. Clearly, the QoS-temperature interplay presents a viable tradeoff for extending the durations of sustainable QoS.

(2) Fast, Analytical DVFS: We identify that, in addition to controlling the average processor frequency for QoS control, fast frequency switching and temporal distribution of high frequency states can significantly improve thermal-efficiency. We realize this novel observation via a kernel-level DVFS state scheduler unit whose principle is illustrated in Figure 4. As a result of processor thermal time constants [11, 20], temperature gradually responds to frequency change. This phenomenon implies that, as illustrated in Figure 4b, breaking the high frequency operation into short intervals and temporally distributing them over time can minimize the peak temperature. Note that both DVFS schedules provide the same average frequency. An analytical proof of this technique for minimizing the peak temperature can be found in our recent work [28]. While the benefits of quick DVFS have been explored only in thermal simulation environments [11, 20], we the demonstrate its effectiveness on a real system for the first time. One critical issue that limits the thermal-optimization via quick DVFS on a real hardware is the performance overhead of switching. This case is pointed out in Figure 5 which shows the measured performance overhead over different switching granularities. Reducing the

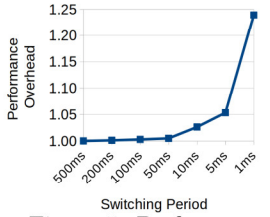

Figure 5: Performance overhead of DVFS.

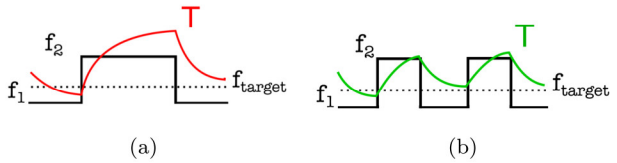

Figure 4: The key idea of the thermally-efficient DVFS state scheduler. By performing quick frequency switching and temporally distributing the states over time (right), the peak temperature can be reduced.

switching frequency beyond 5ms can incur as much as 25% performance overhead.[3] In order to deal with phase variations within the applications as well as the potential changes in the target QoS levels in runtime, on top of the DVFS state scheduler, we designed a PI-type feedback controller for responsively tuning the average processor frequency level towards achieving target QoS levels.

(3) Criticality-driven Scheduling: In our research, we demonstrate that mobile applications are driven by a small number of highly *QoS-critical* threads that determine the user experience and such threads could be detected via a simple offline characterization process. This novel observation is in contrast to prior belief which simply assumed that all the threads of an application are equally important [24, 33]. We propose to use the high-performance cores in a heterogeneous multi-core platform only for accelerating the critical-threads while leveraging the low-power cores in the system for executing the other *non-critical* application threads. Such criticality-driven task scheduling significantly reduces temperatures while still achieving similar performance. Consider the example in Figure 6 that shows, at two different frequencies, the QoS improvement when the threads of a *Rock Player* multimedia player applications are incrementally moved to high-performance cores on a heterogeneous CPU platform [3, 21]. Performance is dictated by a single application thread that provides the peak QoS, which we deem as *QoS-critical* thread. The default HMP scheduler [4] in current smartphones disregards such information and greedily assigns the threads with high CPU demand to power-hungry cores for performance. This results in higher CPU usage on power-hungry cores (26% for this particular application) without any significant improvements in QoS and leads to quick elevation of temperatures. We apply the same characterization to other applications as well to identify critical threads and communicate this information to runtime thread scheduler.

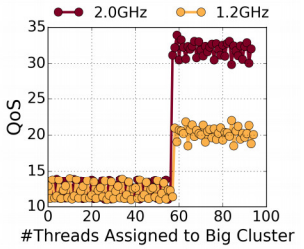

Figure 6: QoS-critical threads of *Rock Player*.

(4) Thermally-Efficient Cores for Execution: We identify significant CPU-GPU thermal coupling in recent SoC architectures that integrate high-performance GPUs. We demonstrate that, depending on the applications' graphics processing demand, thermal efficiency of the of core allocations can alter significantly. For instance, stressing the GPU to peak utilization via an OpenCL microbenchmark [31], Figure 7 shows that the CPU core temperatures can rise by as much as 23°C, *despite the cores being idle*. Not all the cores are effected by the GPU heating at the same rate. For instance, potentially due to close proximity to GPU, the first core heats up the most. We find that, however, the very same core can provide the most thermally-efficient operation when the GPU power is low. Traditionally, CPU is assumed to be the most critical component in mobile system-on-chips (SoC) in terms of thermal-density and thermal management policies have considered CPU cores mostly isolated from the rest of the SoC components. In our research, we monitor the application's GPU usage to determine the most thermally-efficient cores for executing the QoS-critical threads.

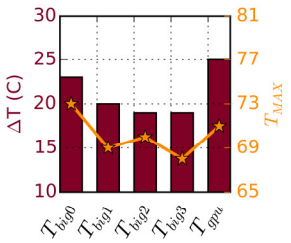

Figure 7: GPU thermal coupling in Exynos 5422.

[3]This experiment is performed on state-of-the-art Qualcomm MDP8974 smartphone.

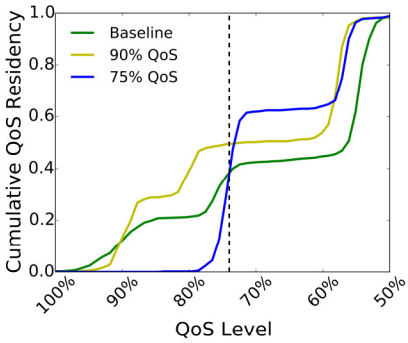

Figure 8: Distribution of QoS for a 10 minute run of Aquarium WebGL application [2] comparing the baseline against the proposed QoS management policy under two different QoS targets. Dashed line is the minimum user requirement.

4. RESULTS AND CONTRIBUTIONS

This sections presents some of the most significant results that we have obtained through real-life implementation of our *thermally-efficient QoS management approach* and explain how they pertain to improving user experience in current generation mobile devices. More exhaustive results can be found in our recent publications [27, 28].

4.1 Results

We implement our techniques and perform investigations on two different mobile platforms; Qualcomm MSM8974 smartphone (Snapdragon SoC, 4-core homogeneous) and an Odroid-XU3 mobile development platform (Samsung Exynos SoC, 8-core heterogeneous (big.LITTLE[3])). To cover a broad range of application domains, we experiment with a wide set of real-life gaming and multimedia applications available for Android as well as with the standard PARSEC [10], SPEC [14] CPU benchmark suites.

'Just enough', yet, sustainable QoS. Figure 8 illustrates how our thermally-efficient QoS management strategy is able to extend the durations of acceptable QoS for a graphics intensive application running on the MDP8974 smartphone. The figure shows the cumulative distribution of the QoS for a 15 minutes continuous execution of the Aquarium [2] WebGL application. A data point corresponds to the fraction of the overall execution time spent above the corresponding QoS level and 100% QoS corresponds to 40 FPS. As pointed by the prior research [23], we consider the 30 FPS (shown by the dashed line) as the bare minimum QoS requirement below which the performance degradation becomes perceptible. The proposed QoS tuning policy with 75% target (30 FPS) increases the sustainability of this QoS level from 40% of the execution time to 62%. This enables *55% longer duration that the user can be provided with an acceptable FPS level while interacting with the application.*

Exploiting hardware heterogeneity for sustainable QoS. Next, we demonstrate our achievements in sustainable QoS on a heterogeneous multi-core CPU available on Odroid-XU3 [5] platform. Our QoS management strategy (named as QScale) effectively minimizes the thermal density of the high-power A15 cores by restricting their use to only highly QoS-critical threads and selecting the most thermally-efficient cores for execution, which are the two novel observations made in this work to improve the thermal-efficiency in mobile devices. Figure 10 plot the sustained QoS durations for 3 policies (QScale versus two baselines) under different QoS targets. In addition to default Android management, as a representative of state-of-the-art policies that only rely on DVFS for QoS management [8, 18, 19], we also compare our approach to *DVFS-only* policy. Our policy consistently provides the longest durations of sustained QoS for all four applications. Remarkably, while the two baseline policies are not able to provide the peak (100%) QoS for more than 10 seconds for the *Bodytrack* (computer vision application) and *Rock Player* (video player) applications due to throttling, our QScale policy sustains the peak QoS for 70 and 140 seconds for these applications, respectively.

To give a more direct insight into the our results and the behaviour of our runtime management strategy, Figure 9 provides the QoS and temperature traces for the *Rain* [6] online animation that we execute within the Chrome browser. We demonstrate the case where the minimum user QoS requirement is the 90% of the maximum. Two baseline policies quickly exhaust thermal headroom and start throttling after reaching 80°C. QoS constraint is violated around 60 seconds. While the default scheduler utilizes all power-hungry cores during execution, QScale only uses them for two QoS-critical

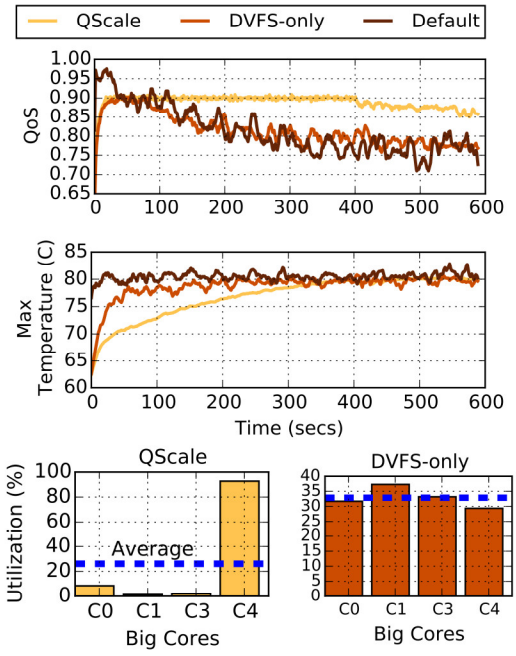

Figure 9: QoS and temperature traces of Rain application for the *default management*, *DVFS-only* and QScale policies under 90% target QoS level. Bottom figures show the average per-core utilization for QScale and *DVFS-only* policies.

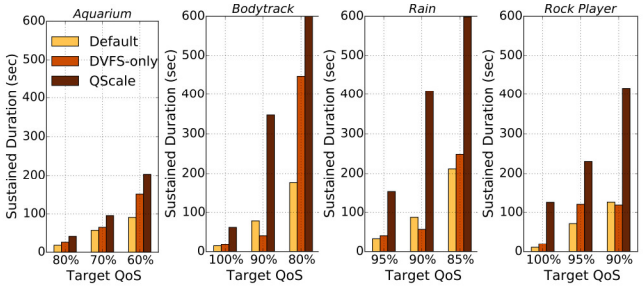

Figure 10: Sustained QoS durations with default management (*Android's Interactive Governor [1] + ARM HMP scheduler [4]*), *DVFS-only* and our QScale policy.

threads and shifts their execution towards thermally efficient cores (C4 and C1) (bottom two figures). As a result, QScale can sustain the QoS above 90% minimum constraint for 400 seconds, giving the user ability to use the application with desired performance for almost 7x longer time.

4.2 Contributions

This research has made several contributions to advance the state-of-the-art in thermal management for smartphones and counteract emerging thermal limitations:

(1) QoS-temperature tradeoff and the need for 'just enough' performance. We have identified a major drawback of existing thermal management approaches that greedily exhaust thermal headroom for performance. Specifically, we have shown that continuously delivering the peak QoS can further accelerate throttling and lead to even more severe QoS loss as users tend to run the applications for longer time. Thus, we motivated the case for 'just enough' performance where we keep the QoS precisely at the bare minimum user acceptable level over long durations for maximizing the durations of *sustainable QoS*.

(2) Novel QoS-centric thermal management techniques for sustainable mobile experience. To provide *thermally-efficient QoS*, we have derived a number of novel observations. Our fast and analytical DVFS management approach aimed at exploiting thermal time constants for lowering temperature. By revealing the thread-criticality in mobile applications and the thermal interactions across the CPU and GPU, we have guided the thread scheduling decisions in highly multi-threaded mobile applications for thermal-efficieny and have shown significant benefits over the current state-of-the-art. Overall, these techniques enable users to interact with their applications for longer durations while still achieving QoS guarantees. Our policies are designed and validated on real-life devices and are purely based user and kernel level software implementations, without relying on any hardware modifications or instrumentations. Thus, our techniques can be readily integrated with default Android distributions and deployed on devices in the wild for larger number of users to benefit from our policies.

5. REFERENCES

[1] Android cpu governors. `http://goo.gl/Mlr9U1`.
[2] Aquarium. `http://goo.gl/2RCoM4`.
[3] big.LITTLE Technology. `http://goo.gl/lMp9QV`.
[4] Heterogeneous multi-processing for big.little. `https://goo.gl/FaYRZG`.

[5] Odroid-XU3 Mobile Development Board. http://goo.gl/2SwDQ1.

[6] Rain. https://goo.gl/5ReAJC.

[7] When benchmarks aren't enough: Cpu performance in the nexus 5. [online] http://goo.gl/ao4kAT.

[8] R. Z. Ayoub et al. Os-level power minimization under tight performance constraints in general purpose systems. In *Proceedings of the 17th IEEE/ACM International Symposium on Low-power Electronics and Design (ISLPED)*, 2011.

[9] A. Bartolini, M. Cacciari, A. Tilli, and L. Benini. A distributed and self-calibrating model-predictive controller for energy and thermal management of high-performance multicores. In *Design, Automation Test in Europe (DATE)*, pages 1–6, March 2011.

[10] C. Bienia, S. Kumar, J. P. Singh, and K. Li. The parsec benchmark suite: Characterization and architectural implications. In *Proceedings of the 17th International Conference on Parallel Architectures and Compilation Techniques (PACT)*. ACM, 2008.

[11] T. Chantem, X. S. Hu, and R. P. Dick. Online work maximization under a peak temperature constraint. In *ISLPED*, pages 105–110, 2009.

[12] B. Egilmez, G. Memik, S. Ogrenci-Memik, and O. Ergin. User-specific skin temperature-aware dvfs for smartphones. In *Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1217–1220, 2015.

[13] M. Halpern, Y. Zhu, and V. Reddi. Mobile cpu's rise to power: Quantifying the impact of generational mobile cpu design trends on performance, energy, and user satisfaction. In *IEEE 22th International Symposium on High Performance Computer Architecture (HPCA)*, 2016.

[14] J. L. Henning. Spec cpu2006 benchmark descriptions. *ACM SIGARCH Computer Architecture News*, 34(4):1–17, 2006.

[15] J. Ho and A. Frumusanu. Revisiting SHIELD Tablet: Gaming Battery Life and Temperatures. [online] Available: http://goo.gl/2hfJaJ.

[16] H. Hoffmann. Jouleguard: Energy guarantees for approximate applications. In *Proceedings of the 25th Symposium on Operating Systems Principles (SOSP)*, pages 198–214, New York, NY, USA, 2015. ACM.

[17] H. Hoffmann, S. Sidiroglou, M. Carbin, S. Misailovic, A. Agarwal, and M. Rinard. Dynamic knobs for responsive power-aware computing. In *Proceedings of the Sixteenth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, pages 199–212, 2011.

[18] D. Kadjo, R. Ayoub, M. Kishinevsky, and P. V. Gratz. A control-theoretic approach for energy efficient cpu-gpu subsystem in mobile platforms. In *Proceedings of the 52nd Annual Design Automation Conference (DAC)*, pages 62:1–62:6, 2015.

[19] D. Kadjo et al. Towards platform level power management in mobile systems. In *27th IEEE International System-on-Chip Conference (SOCC)*, pages 146–151. IEEE, 2014.

[20] M. A. Khan, C. Hankendi, A. K. Coskun, and M. C. Herbordt. Software optimization for performance, energy, and thermal distribution: Initial case studies. In *International Green Computing Conference and Workshops (IGCC)*, pages 1–6, July 2011.

[21] R. Kumar, D. M. Tullsen, P. Ranganathan, N. P. Jouppi, and K. I. Farkas. Single-isa heterogeneous multi-core architectures for multithreaded workload performance. In *Proceedings of the 31st Annual International Symposium on Computer Architecture (ISCA)*, pages 64–, Washington, DC, USA, 2004. IEEE Computer Society.

[22] T. Lanier. Exploring the design of the cortex-a15 processor. *URL: http://www. arm. com/files/pdf/atexploring the design of the cortex-a15. pdf (visited on 12/11/2013)*, 2011.

[23] A. Pathania et al. Integrated cpu-gpu power management for 3d mobile games. DAC, 2014.

[24] A. Pathania, S. Pagani, M. Shafique, and J. Henkel. Power management for mobile games on asymmetric multi-cores. In *IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*, pages 243–248, July 2015.

[25] R. Pozo and B. Miller. Scimark 2.0. *URL: http://math. nist. gov/scimark2*, 2000.

[26] Qualcomm Developer Network. [online] SnapdragonâĎć MSM8974 MDP. https://developer.qualcomm.com/snapdragon-mobile-development-platform-mdp, 2014.

[27] O. Sahin and A. K. Coskun. On the impacts of greedy thermal management in mobile devices. *IEEE Embedded Systems Letters*, 7(2):55–58, June 2015.

[28] O. Sahin, P. Varghese, and A. Coskun. Just enough is more: Achieving sustainable performance in mobile devices under thermal limitations. In *Proceedings of the International Conference on Computer-Aided Design (ICCAD)*. ACM, 2015.

[29] W. Seo, D. Im, J. Choi, and J. Huh. Big or little: A study of mobile interactive applications on an asymmetric multi-core platform. In *2015 IEEE International Symposium on Workload Characterization (IISWC)*, pages 1–11, Oct 2015.

[30] G. Singla, G. Kaur, A. K. Unver, and U. Y. Ogras. Predictive dynamic thermal and power management for heterogeneous mobile platforms. In *DATE*, pages 960–965, San Jose, CA, USA, 2015. EDA Consortium.

[31] J. E. Stone, D. Gohara, and G. Shi. Opencl: A parallel programming standard for heterogeneous computing systems. *Computing in science & engineering*, 12(1-3):66–73, 2010.

[32] Q. Xie et al. Dynamic thermal management in mobile devices considering the thermal coupling between battery and application processor. In *ICCAD*, pages 242–247, 2013.

[33] J. Yun, J. Park, and W. Baek. Hars: A heterogeneity-aware runtime system for self-adaptive multithreaded applications. In *DAC*, pages 107:1–107:6, 2015.

[34] Y. Zhu, M. Halpern, and V. J. Reddi. Event-based scheduling for energy-efficient qos (eqos) in mobile web applications. In *HPCA*, pages 137–149, 2015.

[35] Y. Zhu and V. Reddi. High-performance and energy-efficient mobile web browsing on big/little systems. In *HPCA*, pages 13–24, Feb 2013.