# Mobile CPU's Rise to Power: Quantifying the Impact of Generational Mobile CPU Design Trends on Performance, Energy, and User Satisfaction

Matthew Halpern Yuhao Zhu Vijay Janapa Reddi
The University of Texas at Austin, Department of Electrical and Computer Engineering
{matthalp,yzhu}@utexas.edu, vj@ece.utexas.edu

## Abstract

*In this paper, we assess the past, present, and future of mobile CPU design. We study how mobile CPU designs trends have impacted the end-user, hardware design, and the holistic mobile device. We analyze the evolution of ten cutting-edge mobile CPU designs released over the past seven years. Specifically, we report measured performance, power, energy and user satisfaction trends across mobile CPU generations.*

*A key contribution of our work is that we contextualize the mobile CPU's evolution in terms of user satisfaction, which has largely been absent from prior mobile hardware studies. To bridge the gap between mobile CPU design and user satisfaction, we construct and conduct a novel crowdsourcing study that spans over 25,000 survey participants using the Amazon Mechanical Turk service. Our methodology allows us to identify what mobile CPU design techniques provide the most benefit to the end-user's quality of user experience.*

*Our results quantitatively demonstrate that CPUs play a crucial role in modern mobile system-on-chips (SoCs). Over the last seven years, both single- and multicore performance improvements have contributed to end-user satisfaction by reducing user-critical application response latencies. Mobile CPUs aggressively adopted many power-hungry desktop-oriented design techniques to reach these performance levels. Unlike other smartphone components (e.g. display and radio) whose peak power consumption has decreased over time, the mobile CPU's peak power consumption has steadily increased.*

*As the limits of technology scaling restrict the ability of desktop-like scaling to continue for mobile CPUs, specialized accelerators appear to be a promising alternative that can help sustain the power, performance, and energy improvements that mobile computing necessitates. Such a paradigm shift will redefine the role of the CPU within future SoCs, which merit several design considerations based on our findings.*

## 1. Introduction

Mobile hardware design is driven by ambitious user requirements. Users demand that each generation compute faster, last longer, and fit more components into increasingly thin form factors. The fast pace at which new application use cases, wireless technologies, and sensor capabilities emerge implies that mobile system-on-chip (SoC) designs must quickly adopt and adapt to the rapidly changing conditions, or perish.

At the forefront of this hardware innovation is the mobile CPU. Mobile CPUs are being introduced at an unprecedented rate to keep pace with end-user demands. Fig. 1, based on data mined from over 1700 Android smartphone specifications, conveys the fast pace at which mobile CPU designs have evolved. Considering the ARM-based Cortex-A series alone, the most dominant mobile CPU design in smartphones and tablets to date [1], at least one new CPU core design has been released each year for the last six years – each significantly more advanced than the last. In comparison, x86-based desktop CPU designs did not exhibit as dramatic changes. Intel-based desktop processors only exhibited four significant core design changes throughout the same time span.

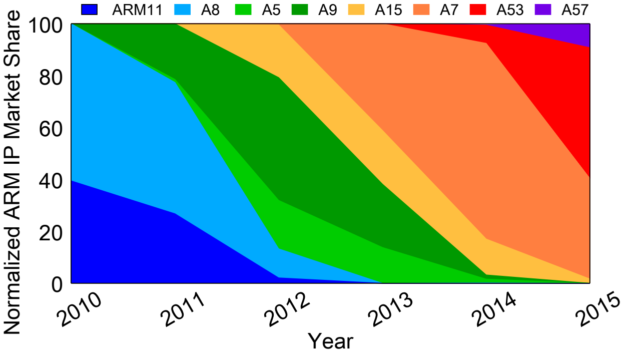


**Fig. 1: Breakdown of yearly ARM Cortex-A CPU design market share. Mobile CPU core designs have rapid design iteration and innovation. At least one new core design is released each year and newer designs overshadow the older ones.**

The rapid design innovation, pervasiveness in society, and power-constrained nature of mobile hardware necessitate the need to understand the implications of their current design trends on future designs. Mobile CPUs have evolved from embedded processors to desktop-like single-chip multiprocessors to provide application responsiveness to end-users. However, mobile CPUs, like any embedded processor, operate under a stricter set of power, thermal and energy constraints than their desktop counterparts. Therefore, there is a need to understand the effectiveness of these designs trends, both in terms of the end-users' satisfaction and hardware efficiency.

In this paper, we take the first steps towards understanding the mobile hardware evolution by studying the mobile CPU in conjunction with end-user experience. We measure and quantify the performance, power, energy, and user satisfaction trends across mobile CPU designs released between 2009 and 2015. Our study spans across ten mobile CPUs, representing the *evolution of the seven consecutive generations of*

978-1-4673-9211-2/16/$31.00 © 2016 IEEE

Table 1: Representative set of Android smartphones and their evolution over the past six years.

| Year | 2009 | 2010 | 2011 | 2012 | | 2013 | | 2014 | | 2015 |
|---|---|---|---|---|---|---|---|---|---|---|
| Manufacturer | Motorola | Samsung | | | | | | | | |
| Name | Droid | Galaxy S | Nexus | Galaxy S 3 | | Galaxy S 4 | | Galaxy S 5 | | Galaxy S 6 |
| Label | D | S | N | S3S | S3Q | S4S | S4Q | S5S | S5Q | S6 |
| SoC | Texas Instruments OMAP 3430 | Samsung Exynos 3110 | Texas Instruments OMAP 4460 | Samsung Exynos 4412 | Qualcomm Snapdragon MSM8960 | Samsung Exynos 5410 | Qualcomm Snapdragon APQ8064T | Samsung Exynos 5422 | Qualcomm Snapdragon 8930AB | Samsung Exynos 7420 |
| Process | 64 nm | 45 nm | | 32 nm | 28 nm LP | 28 nm | 28 nm LP | 28 nm HKMG | 28 nm HPm | 14 nm LPE |
| CPU | ARM A8 | ARM A8 | ARM A9 | ARM A9 | Krait | ARM A15 + A7 | Krait 300 | ARM A15 + A7 | Krait 400 | ARM A57 + A53 |
| Cores | 1 | 1 | 2 | 4 | 2 | 4 + 4 | 4 | 4 + 4 | 4 | 4 + 4 |
| Frequency | 600 MHz | 1 GHz | 1.2 GHz | 1.4 GHz | 1.5 GHz | 1.6 GHz + 1.2 GHz | 1.9 GHz | 2.1 GHz + 1.5 GHz | 2.5 GHz | 2.1 GHz + 1.5 GHz |
| L0 $ (I/D) | - | - | - | - | 4 KB / 4 KB | - | 4 KB / 4 KB | - | 4 KB / 4 KB | - |
| L1 $ (I/D) | 32 KB/ 32 KB | | | | 16 KB / 16 KB | 32 KB / 32 KB | 16 KB / 16 KB | 32 KB / 32 KB | 16 KB / 16 KB | 48 KB / 32 KB |
| L2 $ | 256 KB | 512 KB | | 1 MB | 2 MB | 2 MB + 512 KB | 2 MB | 2 MB + 512 KB | 2 MB | 2 MB + 512 KB |
| RAM | 256 MB LPDDR | 512 MB LPDDR2 | 1 GB LPDDR3 | | 2 GB LPDDR3 | | | | | 3 GB LPDDR4 |
| OS Version | 2.2.3 | 2.2.1 | 4.2.0 | 4.0.4 | 4.1.2 | 4.2.2 | | 4.4.2 | | 5.02 |

*cutting-edge mobile CPU technology*. These mobile CPUs represent eight different microarchitectures, six different process nodes and also include recent trends towards asymmetric multiprocessing and core customization.

Despite almost a decade of existence, mobile CPU design trends and their impact on end-user experience are not well-understood in both industry and academicia. Current mobile CPU architecture research exclusively focuses on the interactions between the hardware and software, largely ignoring the end-user. To extend the conventional research scope to include the end-user, we construct and conduct a novel crowdsourcing-based user study that spans over 25,000 participants using Amazon's Mechanical Turk service. Our methodology allows us to *quantitatively determine the relationship between end user satisfaction and mobile CPU design evolution.* The survey participants evaluate a wide variety of applications that exhibit different types of user interaction and computational characteristics common to current (e.g. Angry Birds), and likely future (e.g. augmented reality), mobile applications.

Our quantitative analysis exposes how mobile CPU design trends have impacted the end-user, hardware design, and the holistic mobile device. To the best of our knowledge, our study is the *first* of its kind to rigorously evaluate mobile CPU design trends. Furthermore, our work can serve as an example for future, more holistic studies that consider the rest of the SoC and mobile device. Each of our mobile CPU observations is quantitatively reinforced – valuable in and of itself – regardless of whether it aligns with conventional wisdom or is surprising:

- **Desktop-like Scaling:** Mobile CPUs have adopted many of the high-performance mechanisms found in desktop CPUs. User satisfaction is latency-sensitive, which emphasizes the need for single-threaded performance improvements. However, the "low hanging fruit" (i.e. low-power) performance-oriented techniques are already being used in mobile CPUs. Future hardware and software will need to understand how to identify and efficiently mitigate user-critical bottlenecks.
- **Multicore CPUs:** Even though multicore CPUs are often under-utilized in mobile applications [31, 32], they serve important roles to deliver end-user satisfaction. User critical application functionalities are often multithreaded to leverage multicores speedups and background processes can interfere with user-facing application processes when there are not enough cores available.
- **CPU Criticality:** Mobile applications are developed in general-purpose programming languages that primarily target the mobile CPU. Even for applications that utilize other SoC components, such as the GPU and image decoder, end-user satisfaction still depends on CPU performance. Therefore, mobile CPU design remains relevant as hardware acceleration and heterogeneous execution catch on.
- **Power Wall:** In contrast to other smartphone components, such as the display and radio, mobile CPU power consumption has risen excessively over time. Single-core power consumption has hit a power wall, and multicore power consumption can significantly surpass SoC-level TDPs without even considering the rest of the SoC. SoC and mobile device designers must pay closer attention to CPU power consumption and consider synergistic, cross-layer power optimizations that fairly allocate power budgets across the mobile CPU and other smartphone components.

The remainder of the paper is organized as follows. Sec. 2 analyzes recent mobile CPU design trends. Sec. 3 presents our crowdsourced user study analysis. Sec. 4 and Sec. 5 discuss the implications of our findings, focusing on CPU scaling design methodologies and specializaton, respectively. Prior work is discussed in Sec. 6, and we conclude in Sec. 7.

## 2. Mobile CPU Evolution

Today's desktop and server CPUs are the result of generational microarchitectural enhancements, clock frequency increases, memory hierarchy growth and multicore scaling. Mobile CPUs are no exception, having embraced these architectural design features at an unprecedented pace in pursuit of performance. Through measurement, we quantify their impact on performance, power and energy across seven mobile CPU generations, released from 2009 to 2015. Our study focuses on peak performance because it drives design innovation. We study mobile CPUs that incorporate the cutting-edge mobile CPU technologies introduced each year (Sec. 2.1).

Our measured results allow us to make several key obser-

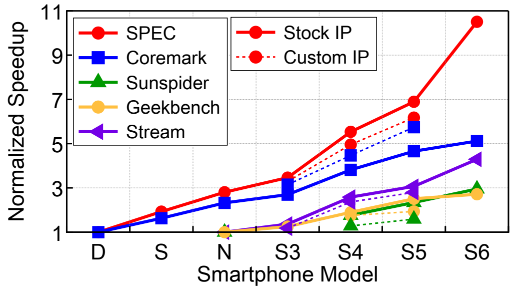


**Fig. 2: More aggressive mobile CPU design techniques have provided significant performance improvements over time.**

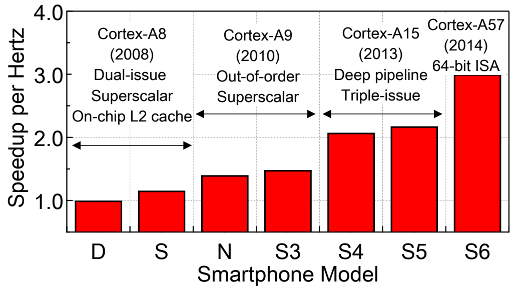


**Fig. 3: Microarchitectural innovations alone have enabled substantial performance improvements across the mobile CPUs.**

vations about the current state of mobile CPU design. Mobile CPU designs provided substantial performance improvements generation-after-generation by rapidly adopting desktop level design techniques at an unprecedented pace (Sec. 2.2). However, these improvements have come at the expense of increasingly higher power consumption. Moreover, while energy-efficiency improved rapidly during the early years, improvements have diminished in recent years (Sec. 2.3).

### 2.1. Mobile CPU Generations

An important aspect of conducting any generational study is selecting the right "samples" to study. Our work focuses on ten ARM-based mobile CPUs released between 2009 and 2015. We use "CPU" to refer to the all of the processing subsystems that support general purpose compute (i.e. core and memory). Both the core and memory subsystems have dramatically improved over time, so we study their holistic evolution across the mobile device generations.

The mobile CPUs we study, shown in Table 1, capture the rapid mobile CPU design innovation exhibited over the last seven years. Each mobile CPU is found within a top-selling smartphone that encompasses the cutting-edge technology available for that particular year, tracking the mobile CPU adoption trends in Fig. 1. Other mobile devices also use these CPU designs. For example, both the Samsung Galaxy Tab 12.6 tablet and Samsung Galaxy S5 (`S5S`), and the Google Glass and Samsung Galaxy Nexus (`N`), utilize the same system-on-chip (SoC) families. Throughout the rest of the paper, we refer to each smartphone model by its label abbreviation. The ten mobile CPUs span seven different microarchitectures, five major process technology nodes, different cache configurations and memory technologies, and also multicore designs.

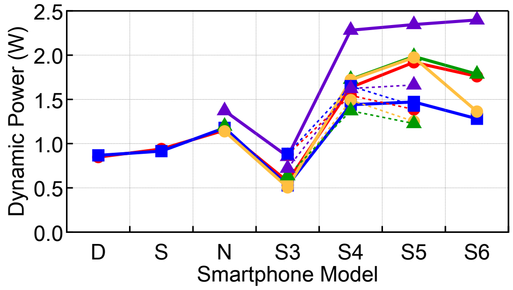


**Fig. 4: Mobile CPU core power consumption appears to saturate around 1.5 W, suggesting later designs hit a "power wall".**

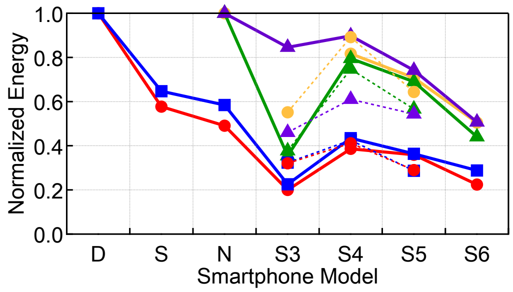


**Fig. 5: Energy efficiency gains have begun to slow as power consumption increases outweight performance gains.**

For completeness, we also consider different design methodologies between the various CPU manufacturers. We study two CPUs vendor designs for each year from 2012 to 2014. Samsung uses *stock* ARM A7 and A15 microarchitectures in a heterogeneous multicore configuration whereas Qualcomm creates its own *custom* microarchitecture (Krait) and homogenous multicore CPU for the ARM instruction set architecture. The Krait, used in the `S3Q`, `S4Q`, and `S5Q`, provide an example for a "custom" ARM-based processor design. The core design is similar to the A15 with a triple-issue out-of-order pipelines but with a more tightly-integrated cache design and a shorter pipeline depth. Nonetheless, each Krait design can operate at higher clock rates than the stock design.

### 2.2. Mobile CPU Performance Trends

We use industry standard CPU-intensive benchmarking applications to isolate the peak single-core performance of each CPU. These benchmarks are well-established in both industry and research. We address interactive mobile applications (Sec. 3) and various power management mechanisms (Sec. 4) later in the paper. All of the benchmarks are compiled *statically* with `gcc 4.5.2` to be robust to the devices' different OS kernel versions. However, it was too old to support the `S6`'s ARMv8-a architecture so we use `gcc 4.8.3` instead.

We represent embedded benchmarking with EEMBC's `Coremark` benchmark, which is well-established within the embedded market segment. The benchmark has been used in prior research to evaluate mobile CPUs [26], as well as in industry white papers [2]. More recently, `Geekbench` [3] has emerged as a popular mobile benchmarking suite and Sunspider [4] is the de facto JavaScript/Web benchmarking suite to date. We also include SPEC CPU 2006 benchmarks [5]. Various industry partners, spanning different companies acknowledge the use of SPEC CPU to evaluate mobile CPU designs [36, 46, 49]. We use a subset of the benchmark (gcc, libquantum, omnetpp, hmmer, and bzip2 – input_program). Memory limits force us to use the train inputs, facing similar issues as [26]. In addition, compiler and workload memory footprint issues limit other CPU 2006 benchmarks and workloads from being run on the earlier systems. A data point will be absent for these rare cases.

ARCH OBSERVATION: #1: *Mobile CPUs have achieved a 10X performance improvement in a seven-year time span by rapidly adopting design techniques used in desktop CPUs.*

Fig. 2 shows the single-core speedup for CoreMark, SPEC, Sunspider, Geekbench and Stream workloads. The data is presented relative to D, the oldest phone in our study, and smartphones become more recent in the rightward direction along the *x*-axis. The solid lines represents the stock ARM IP line (e.g. Samsung and TI) and the dashed lines denote the custom ARM IP (e.g. Qualcomm). The S6, the newest device, achieves a 10X average speedup over D for CoreMark and the SPEC workloads. On average, performance approves 32% generation-to-generation.

Frequency scaling has fostered significant performance improvements across mobile CPU generations. As Table 1 shows, clock frequency increased by over 4X (500 MHz per year). In 2009, the D operated at 600 MHz, whereas the S5Q reached a top clock frequency of 2.5 GHz in 2014 – near PC speeds.

Performance improvements cannot be contributed to frequency scaling alone. Fig. 3 shows the performance of the seven stock CPU designs normalized by their corresponding clock frequency. Microarchitecture-level and the memory hierarchy improvements were able to provide an almost 3X speedup from D to the S6, without considering frequency.

The oldest phones we study, the D and S, use the A8 (2008). Unlike its predecessor, the single-issue ARM11, the A8 has a dual-issue in-order superscalar design [6] to exploit instruction-level parallelism. The transition for in-order to out-of-order pipeline designs facilitated significant performance improvements. The A9 (2010), used in the N and S3, utilizes a dual-issue out-of-order pipeline [6]. Even more aggressive, the A15 (2013), utilized in the S4S and S5S, increases the depth and issue width of its out-of-order pipeline beyond the A9 [7]. The A57 (2014), used in the S6, incorporates a new 64-bit instruction set architecture (ISA) into an A15-like design [8].

On-chip and off-chip memory hierarchy enhancements also facilitated performance improvements. The most recent S6 incorporates a larger 48 KB L1 instruction cache to address the growing instruction footprints of mobile applications [44] while the L1 data cache size remains fixed at 32 KB. Beyond the D, mobile CPUs incorporated a shared L2 cache, which also double in size from 512 KB at the S to 1 MB at the S3S to 2 MB at the S3Q for the remainder of the CPUs. Off-chip DRAM also evolved to support the CPUs. From LPDDR to LPDDR4, data rates doubled from one generation to the next, starting at 400 MHz and reaching 3.2 GHz.

### 2.3. Mobile CPU Power and Energy Trends

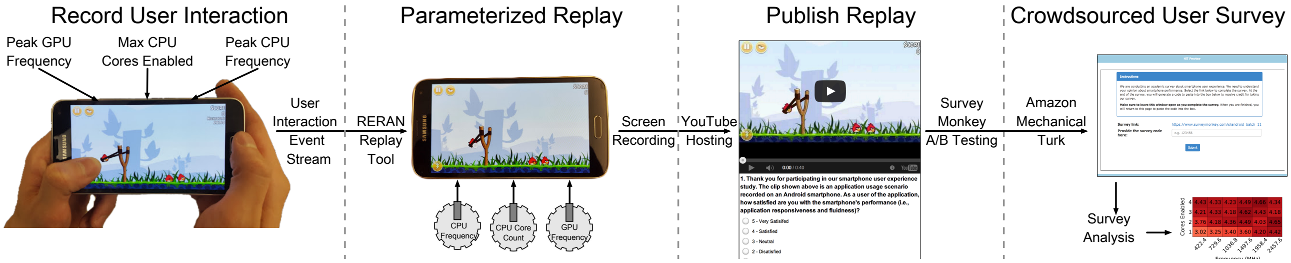
Performance improvements have come at the expense of power and energy consumption. Smartphones do not provide (or openly disclose) mechanisms to directly measure CPU power consumption. Instead, we use differential power measurement techniques practiced in prior work [26] to extract dynamic power consumption. Battery-level power measurements are collected from each device using the Monsoon Power Meter [9], which has a sampling rate of five kilosamples per second and performs self-calibration. We use differential power measurements ($P_{active} - P_{idle}$) to isolate the CPU's dynamic power consumption and remove static power consumption from the idle and unused components (e.g. display, radio, GPU). We disable the radio and other components unrelated to our study before each power measurement experiment.

ARCH OBSERVATION: #2: *The mobile CPU's single-core thermal design point (TDP) has saturated at around 1.5 W, similar to the 100 W power ceiling common to desktop CPUs.*

Fig. 4 shows the power consumption trend across mobile CPU generations. Initially, power consumption mostly reduced as performance improved from the D's in-order A8 design to the S3S's out-of-order A9 design. The power consumption for all of the workloads reduced from 0.8 W to 0.5 W (38%). However, S4S begins a trend where complex coupled with higher clock frequencies increases have caused the average power consumption to hover around 1.5 W. We observe this trend for the five most recent smartphone generations. At its peak, the S5S's power consumption almost reaches 2 W during SPEC's execution. Somewhat similar behavior is observed during experiments in the most recently released S6.

Stream exemplifies the different design strategies for the stock and custom ARM cores. Fig. 2 and Fig. 4 demonstrate that the custom Krait cores pursue performance improvements that are more power-efficient than the stock cores. The S5S scores 10% higher than the S5Q in performance but does so with almost 50% higher power consumption because of its more aggressive pipeline and memory hierarchy subsystem.

Process technology has played a large role in curbing power consumption. When the A9 shrank from 45 nm in N to 32 nm in S3S, power consumption dropped by 44%. The S3S was fabricated using the high-k metal gate (HKMG) technology, which utilizes a new gate-level dielectric to minimize static leakage. The remaining CPUs also use processing nodes with HKMG technology (or one of the LP and HPm variants). HMKG is a prime example of "good [and rare] fortune" in processor evolution [45]. Process innovations do not occur frequently, so we do not see large improvements (or dips) in



1. Thank you for participating in our smartphone user experience study. The clip shown above is an application usage scenario recorded on an Android smartphone. As a user of the application, how satisfied are you with the smartphone's performance (i.e., application responsiveness and fluidness)?

○ 5 - Very Satisifed

○ 4 - Satisfied

○ 3 - Neutral

○ 2 - Disatisfied

We are conducting an academic survey about smartphone user experience. We need to understand your opinion about smartphone performance. Select the link below to complete the survey. At the end of the survey, you will generate a code to paste into the box below to receive credit for taking our survey.

**Make sure to leave this window open as you complete the survey.** When you are finished, you will return to this page to paste the code into the box.

**Fig. 6: Our crowdsourced user study methodology. (a) We record a golden user interaction trace. (b) We deterministically replay the user interaction trace with various CPU (and GPU) configurations and record a usage video clip. (c) We package the video clips into a randomized survey. (d) We post the survey to Amazon Mechanical Turk and ask the participants to rate the playback.**

CPU power consumption in the following generations.

ARCH OBSERVATION: #3: *Mobile CPU energy efficiency improvement plateaued as the performance benefits do not sufficiently make up for the additional power consumption.*

Fig. 5 shows CPU energy consumption across the six generations normalized to D. We observed rapid energy efficiency improvements between D and S3S. Simultaneous performance improvements and power reductions reduced single-core energy use by as much as 80%. For the next two mobile CPU generations (S4S and S4Q), energy efficiency worsens as these mobile CPUs are unable to sustain performance improvements without sacrificing power efficiency. The S5S and S5Q almost double the S3S's energy consumption. Qualcomm's custom core designs consume less power than their Samsung-manufactured counterparts, but also typically lag in performance. The Qualcomm core's power-efficiency outweighs Samsung's performance advantage to provide better energy-efficiency. Finally, the S6 achieves substantial performance improvements beyond the S5S and S5Q without further increasing power consumption. Thus, it is capable of achieving energy efficiency almost on par with the S3S.

## 3. Bridging CPU Design and User Satisfaction

Mobile CPU designs have evolved tremendously over time to satisfy end-users. Unfortunately, the performance enhancements have come at the expense of excessively high power consumption. *The goal of this section is to quantitatively capture the relationship between these power-hungry mobile CPU advancements and end-user satisfaction.* Specifically, we:

1. separate the contributions of single- and multicore performance advancements on achieving end-user satisfaction
2. quantify the degree to which mobile CPUs provide end-user satisfaction and whether future improvements are needed
3. determine the mobile CPU's role and impact on end-user satisfaction amongst accelerators in a system-on-chip.

A rigorous methodology for quantifying end-user satisfaction does not currently exist in computer systems research, so we construct and conduct a new methodology for our study (Sec. 3.1). We leverage the notion of crowdsourcing to conduct a user study spanning 25,478 participants, whom we solicited through the Amazon's Mechanical Turk service [10]. Our large-scale user study allows us to comprehensively assess mobile users' sensitivities to different CPU architecture and performance configurations with high statistical confidence.

We study a broad range of interactive mobile applications that span different application domains and also exhibit different computational characteristics. Our results demonstrate the role mobile CPU improvements played in improving end-user satisfaction. We show how mobile CPU design trends have enabled more advanced mobile applications and made them satisfactory to end-users over time. Almost all of the applications we study can achieve user satisfaction, but they require different amounts of single- and multicore performance. Our data allows us to determine quantitatively the degree to which mobile CPU enhancements have been worthwhile in the midst of high power consumption (Sec. 3.2). The applications that do not provide satisfactory experiences suggest that mobile CPUs designs will need to evolve further despite power limits.

### 3.1. Mechanical Turk-Based User Study Methodology

Our crowdsourced study consists of participants ranking their satisfaction while we replay representative application use cases under various CPU performance configurations, i.e., core counts and clock frequency. Our study showcases a new approach to conducting architectural-user studies at scale.

**Mechanical Turk** Amazon offers Mechanical Turk (MTurk) [10], which is an Internet marketplace for Human Intelligence Tasks (HITs). Requesters post tasks with a price and solicit "workers" to perform the tasks.

Mechanical Turk is a popular mechanism for conducting crowdsourced experiments. It has been used successfully in other research areas, such as for computer vision training data [27] and answering psychological questionnaires [27].

We solicited 25,478 users for our study. We got high user engagement by posting $0.10 HITs for workers. Each configuration within an application is scored by *at least 50 participants* to provide statistical confidence in our results. Fig. 6 summarizes our MTurk-based experimental workflow.

**Applications** We study a broad range of popular Android applications, shown in Table 2. Our application selection criteria decompose applications beyond typical application domain categories into user- and hardware-level metrics.

Our user-oriented application selection criteria include various user behaviors (e.g. waiting for a webpage to load,

**Table 2: Application descriptions, use cases and observed computational diversity.**

| | Application Description | | User-level Metrics | | | Computational Metrics (TLP) | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | **Name** | **Description** | **Installs** | **Duration** | **Events** | **1** | **2** | **3** | **4** | **Avg** |
| **Current-Gen** | `Angry Birds` | Navigate to and play first level | 0.5-1E9 | 0:41 | 6 | 21% | 8% | 2% | 0% | 1.43 |
| | `CNN (Chrome)` | Navigate to and scroll through CNN.com | 1-5E8 | 0:36 | 12 | 16% | 11% | 7% | 2% | 1.90 |
| | `Epic Citadel` | Navigate through environment | 0.5-1E6 | 0:44 | 15 | 25% | 22% | 5% | 0% | 1.67 |
| | `Facebook` | Log-in and visit ESPN brand page | 0.5-1E9 | 0:57 | 23 | 16% | 8% | 3% | 1% | 1.67 |
| | `Gladiator` | Sword-fight opponent in first level | 1-5E6 | 0:36 | 31 | 31% | 8% | 2% | 0% | 1.34 |
| | `Photoshop Express` | Apply various filters and effects to image | 1-5E7 | 0:48 | 15 | 13% | 9% | 6% | 15% | 2.52 |
| | `Youtube` | Navigate to and watch video | 1-5E7 | 0:46 | 13 | 16% | 10% | 5% | 1% | 1.73 |
| **Next-Gen** | `Ambiant Occlusion` | Brute force ray primitive intersection | 1-5E3 | 0:21 | 4 | 7% | 3% | 2% | 46% | 3.46 |
| | `Face Detection` | Face detection on video | 1-5E3 | 0:21 | 3 | 17% | 4% | 2% | 47% | 3.09 |
| | `Gaussian Blur` | Guassian Blur on video | 1-5E3 | 0:21 | 3 | 51% | 4% | 2% | 4% | 1.37 |
| | `Julia` | Visualization of Julia Set dynamics | 1-5E3 | 0:17 | 4 | 11% | 4% | 2% | 24% | 2.93 |
| | `Particles` | Particle simulation in a spatial grid | 1-5E3 | 0:21 | 4 | 17% | 14% | 14% | 7% | 2.21 |

watching a video, etc.). To convey the variety of interactiveness across applications, we present the number of interactive events (e.g. tap, swipe, etc.) used to exercise each application use case in the "User Events" column.

The application use cases also exhibit diverse computational characteristics. We measure each application's thread-level parallelism (TLP) with the systrace Android utility to identify the amount of parallelism hardware can exploit [30].

We also incorporate applications from emerging application domains, such as augmented reality and physics simulation. These forward looking applications are are part of CompuBench [11], an industry-strength benchmark suite, used by various mobile device manufacturers [38]

**Record User Interaction** For each application, we record a user manually performing a representative use case. The Android `getevent` utility captures raw touchscreen driver events that capture user input and timing seen throughout the user interaction. To ensure reproducibility of these interactive "use cases" during later replay stages, we use the `RERAN` [33], which is a low-overhead, deterministic touchscreen event injection tool for the Android platform.

We record each application on the `S5Q` operating at its peak performance (i.e., all four CPU cores at 2.4 GHz).[1] We deem this the baseline user interaction trace because it maximizes application responsiveness on the device, which in turn maximizes the likelihood of achieving end-user satisfaction [42].

**Parameterized Replay** To study the impact of mobile CPU evolution on user satisfaction, we replay the interactive use cases while we sweep `S5Q` single- and multicore performance configurations. The device's power management facilities (e.g., DVFS) are disabled to ensure the clock frequency and the number of enabled cores remains fixed throughout each replay session. By parameterizing single- and multicore performance across the `S5Q`, from the latest CPU generation, we can simulate the CPU performance configurations found across the earlier mobile CPU generations we study. We conduct a rigorous performance analysis to find the `S5Q` performance configurations that correspond to the peak performance of the earlier mobile CPU generations. Sec. 3.2 provides additional details and validation of this method.

**Publish Replay** During each replay session, we record a video clip using `screenrecord` to include in our survey. We host the recorded video clips of the different processor performance configurations on `youtube.com`.

**Crowdsourced User Survey** We conducted our user study through surveys on `surveymonkey.com`. Each user satisfaction survey consists of a single, randomly selected video clip and multiple choice question that asks the user to rate their satisfaction of the video. We ask, "how satisfied are you with the smartphone's performance (i.e., application responsiveness and fluidness)?" We provide five simple answer choices common to many satisfaction surveys: (1) Very Dissatisfied, (2) Dissatisfied, (3) Neutral, (4) Satisfied, and (5) Very Satisfied.

**Due Diligence and Validation** We took several steps to validate our crowdsourcing user methodology. Before posting our survey, we watched all of the videos to make sure there were not any errors during the recording phase. Also, we also evaluated the videos across a small group of users in-house which proved to be consistent with the trends we observe across our Mechanical Turk participants.

Overall, we observed that our participants had good intentions for our survey. Studies that have scrutinized crowdsourcing have quantitatively shown this to be true as well [39]. As a part of our results, we collected the response time for each user. Users remained in the survey long enough to have watched their assigned video. Only a negligible few (< 1%) abandoned the video or survey, and as such they do not affect our results.

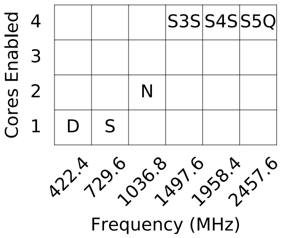


**Fig. 7:** `S5Q` **CPU mapping.**

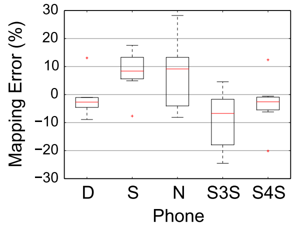


**Fig. 8: Mapping error.**

[1] The `S6`'s release date overlapped with our ongoing crowdsourcing experiments. But since its availability, we have rerun experiments on the `S6` when it is necessary to reinforce a general conclusion we draw (e.g., as in Sec. 2).

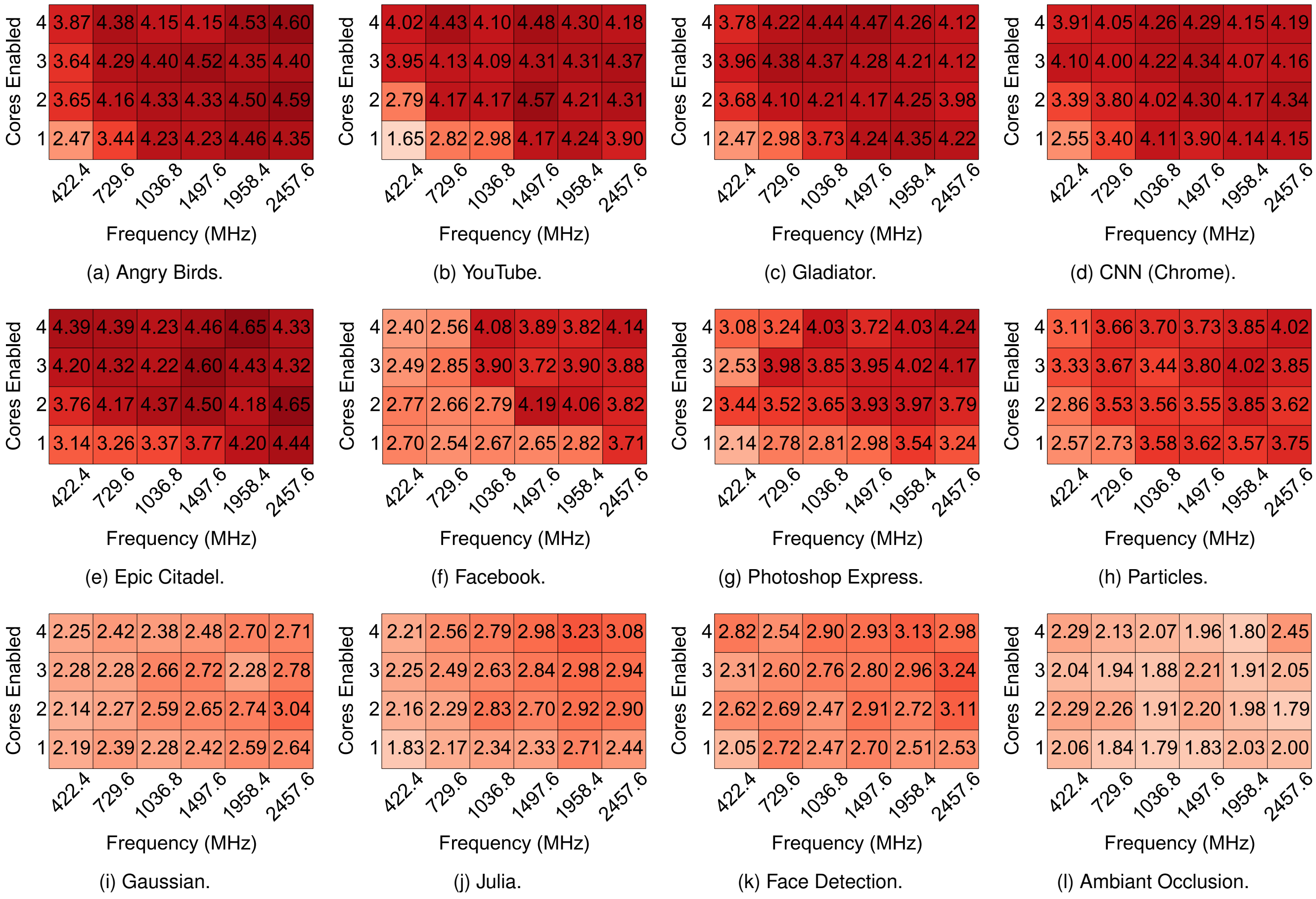


**Fig. 9: User satisfaction across single- and multi-core parameter sweep on** `S5Q`**. Rating system: (1) Very Dissatisfactory (2) Dissatisfactory (3) Neutral (4) Satisfactory (5) Very Satisfactory. Results: Only tiles whose satisfaction score differ by more than 0.52 should be compared.**

### 3.2. User Satisfaction and Architecture Implications

We present the results of our crowdsourced user study for the workloads in Fig. 9. Each heatmap corresponds to an application in Table 2. The heatmap cells represent the user satisfaction score for a particular (single-core, multicore) performance configuration that increases along the *x*- and *y*-axis, respectively. The intensity of a tile corresponds to the average satisfaction score. The darker the tile, the more satisfactory the application use case was with that performance configuration.

To form sound conclusions between adjacent tiles, we determined the confidence interval for each configuration. On average, the 95% confidence interval for each configuration extends 0.26 from the reported average score centered in the tile. Thus, only tiles whose satisfaction score differ by more than 0.52 should be compared. For example, in Fig. 9a it is reasonable to conclude that that user satisfaction improves from (729.6 MHz, one core) to (1036.8 MHz, one core). However, the same conclusion cannot be reached by comparing (1036.8 MHz, one core) to (1958.4 MHz, one core).

To allow intuitive comparison between different mobile CPU generations, we map the performance of earlier smartphones to `S5Q`. Specifically, the peak single-core performance of each earlier phone is mapped to an `S5Q` DVFS frequency that provides the cloest performance. Multicore performance is approximated with CPU core count. Fig. 7 shows the mapping. Using `S3S` as an example of reading the mapping, its location indicates that its peak single-core performance is closest to 1497.6 MHz on `S5Q`, and it has 4 cores. Fig. 8 shows that the mapping error is less than 10% for each phone.

<u>USER OBSERVATION: #1:</u> *User satisfaction is latency-sensitive, and as such gigahertz clock frequencies and out-of-order pipelines provide the single-threaded performance improvements needed to achieve high end-user satisfaction.*

Early mobile CPU designs struggled to provide sufficient single-threaded performance. None of the tiles corresponding to the single-core in-order A8 CPUs found within the `D` and `S` were "satisfactory" to survey participants. In interactive gaming, such as Angry Birds (Fig. 9a) and Gladiator (Fig. 9c), and webpage loading (Fig. 9d), users expect faster response times. The transition to the out-of-order A9, used in `N` and `S3S`, makes these applications satisfactory. Although `CNN (Chrome)` and `YouTube` (Fig. 9b) each has a thread-level parallelism (TLP) [30] close to two (Table 2), a single core A9 achieves satisfactory user experience for them.

More aggressive out-of-order core designs were needed to meet the response latencies end-users expect for other applications. For example, Epic Citadel (Fig. 9e) uses the computationally intensive Unreal Game Engine [12]. Single-core performance on par with an S4S A15 core can provide a satisfactory experience to participants.

USER OBSERVATION: #2: *Multicore mobile CPUs contribute to end-user satisfaction improvements because they support emerging multithreaded mobile applications more effectively than a single high-frequency core, tolerate worst-case application activity bursts more gracefully and improve application performance by mitigating shared resource contention.*

The proliferation of multicore mobile CPUs have helped achieve user satisfaction improvements for several reasons. First, some applications rely on multicore capabilities by design. Multimedia applications, such as Photoshop, leverage data-level parallelism within signal processing algorithms to enable multithreading. Photoshop (Fig. 9g), has a TLP of at least three for 48% of its non-idle runtime. As a result, it requires multiple cores to deliver a satisfactory experience. It first becomes satisfactory at four cores with the performance of an N core. Similarly, Particles (Fig. 9h), whose average TLP is 2.21, requires at least three S3S cores.

Second, multicore CPUs can alleviate worst-case application interaction bursts that threaten otherwise high user satisfaction. For example, Facebook (Fig. 9f) requires at least two cores to provide end-users satisfactory responsiveness while logging into the application. Login is a bursty and multitasking application process. The application must process network requests to retrieve application content and then render it on screen. While substantial computational resources may not be needed for steady-state application usage scenarios, application launches, and logins are well-established application use cases that can impact user satisfaction [58]. To provide the same level of user satisfaction, the S5Q would have to run at peak single core frequency, but even then the result is only marginally satisfactory to users.

Third, multicore CPUs mitigate the contention between application and background threads that can affect user experience. Gladiator has the least TLP of all applications (1.34). Its performance relies heavily on the CPU's single-thread execution capabilities. On the S5Q, the application needs to run at nearly 1.5 GHz when one core is enabled. However, similar high user satisfaction can be achieved by cutting the frequency by half and running at 729.6 MhZ using two cores. Background tasks that interfere with the main thread's execution are readily offloaded by the kernel to the second core, allowing the first core to operate undisturbed.

USER OBSERVATION: #3: *Single- and multicore mobile CPU performance improvements are still needed to achieve end-user satisfaction for emerging application domains that rely on high-performance parallel programming frameworks.*

With the proliferation of multicore processors in recent years, there has been growing interest in supporting computationally challenging applications efficiently through the use of parallelism. Many parallel programming frameworks, such as Mare [13], RenderScript [14], and OpenCL [15], are emerging to support general-purpose computation on mobile platforms.

We evaluate several forward-looking applications from emerging application domains, such as perceptual computing, augmented reality, and advanced image processing. These applications are built using the RenderScript framework and targeted specifically at mobile multicore CPUs. The applications are much more computationally intensive than the mainstream applications. Most of them spend a significant amount of non-idle execution time on all four cores. Their average TLP is 2.41. User events in these applications is low because they do not require heavy interactivity to use.

We find that these next-generation applications require single- and multicore improvements beyond what today's mobile CPUs provide. For instance, Gaussian Blur has high single-threaded performance requirements. It spends the majority of its non-idle execution time executing within a thread. With an average TLP of 1.37, Gaussian Blur does not see a dramatic satisfaction improvement as more cores are added at peak frequency (Fig. 9i). Julia (Fig. 9j) with average TLP of 2.93 and 14% of execution time with a TLP of four, sees satisfaction increase from unsatisfactory to neutral as it maximizes resource utilization. However, enough end-user satisfaction (> 4.0) has still not been achieved.

To validate that our participants are capable of recognizing satisfactory performance for these applications, we conducted the survey a second time based on a desktop system. Our participants noticed a dramatic user experience improvement and declared them as satisfactory, which implies satisfaction *is* in fact attainable for these workloads for our survey participants.

Furthermore, we ran the crowdsourcing experiments a third time to confirm single- and multicore performance improvements beyond the S6 are needed in future mobile CPUs. Recall that we use the S5 for our experiments. The S6 was unavailable at the time of our experiments. Despite the performance enhancements in the S6, we observed similar results as we did with the S5Q. User experience was unsatisfactory.

USER OBSERVATION: #4: *Even for applications that utilize other SoC components, such as the GPU and image encoder/decoder heavily, mobile CPU performance capability remains critical for achieving high end-user satisfaction.*

Mobile applications typically rely on a variety of on-chip SoC accelerators to provide rich end-user experiences, and this trend will likely continue into the future. Therefore, there is a need to understand the extent to which the mobile CPU impacts user satisfaction. All mobile applications exercise the mobile GPU to some degree, making it the most heavily utilized SoC accelerator. We found that user satisfaction was not sensitive to performance differences across the S5Q's Adreno 330 GPU for almost all of the applications we studied. Fig. 10 shows user satisfaction as we sweep the S5Q's Adreno 330 GPU frequency while the CPU operates with all four cores at peak frequency. Besides Gladiator, user satisfaction does not significantly change as GPU frequency increases from 200 MHz to 578 MHz. Gladiator is the most aggressive interactive use case we study, with 31 user events in a 36 second timespan, spawning a significant number of screen updates. Thus, user satisfaction increases with frequency by nearly fourfold from the lowest to highest GPU frequency. GPU computations invoked by the other current-generation applications are infrequent and underwhelming compared to the CPU-based computation. The forward-looking applications are too compute bound and CPU-stifled to stress the GPU.

Applications also rely on fixed-function acceleration. Multimedia applications, such as YouTube and NetFlix, rely on specialized hardware accelerators to to avhieve high frame rates. For instance, YouTube by default uses the VP9/WebM video coding format, used in the S5Q. However, the CPU remains on the critical execution path even though computations are offloaded to these accelerators. Fig. 9b shows that if the single-core CPU performance drops below 1.5 GHz, user satisfaction plummets from 4.17 to 2.98. The is because mobile CPU has to manage the device drivers to use these accelerators while also orchestrating other computations [57].

## 4. The Limits of Desktop-like CPU Scaling

In the previous section, we demonstrated that the CPU plays an important role in mobile devices. User-critical application functionalities execute on the CPU, and its performance greatly impacts end-user satisfaction. The CPU will play a key role in the next generation of mobile computing applications, which will require performance improvments to be sustained across future mobile CPU generations. However, mobile devices are both thermal- and energy-constrained, yet mobile CPUs power consumption continues to increase without adequate cooling and battery advancements (Sec. 4.1).

Our measurements quantitatively demonstrate that mobile CPU designs are approaching the "power wall", similar to what has been observed in desktop CPU design. By comparing design trends of mobile and desktop CPUs, we attribute CPU scaling methodologies (e.g. larger caches, higher frequencies, more cores, etc.) that have been successful in desktop being unsustainable in mobile CPUs due to the lack of power scaling improvements in recent process technologies (Sec. 4.2). Our measurements quantitatively reinforce recent works [28, 55], that have that have also projected the demise of CPU scaling in CPU design. These results suggest that mobile CPU designs need to be optimized in fundamentally different ways than past designs have been.

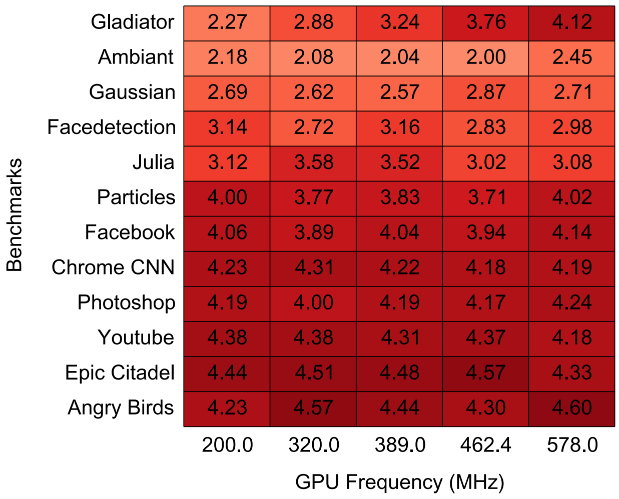


**Fig. 10: Satisfaction score across mobile GPU performance scaling. Only tiles whose score differ by more than 0.52 should be compared. Most mainstream applications are largely unaffected by the GPU's performance, and instead they are more sensitive to the CPU (Fig. 9).**

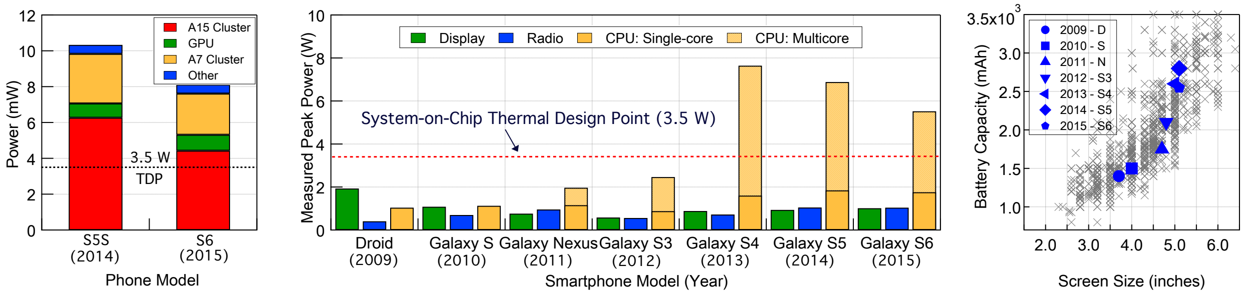
### 4.1. System-level Design Constraints

Mobile CPU designs are constrained to operate under strict thermal design points and energy budgets. While both these limits also apply to desktop and server CPUs, these limits are much more severe for mobile CPUs. Understanding the severity of these limits gives us important insights on alternative approaches for designing future mobile CPUs.

**Thermal** Mobile CPU performance is primarily limited by thermal constraints. However, mobile CPU manufacturers are creating mobile CPUs that can exceed their sustainable thermal power budgets with the assumption that they will not be utilized to their peak capabilities. Given the fact that mobile applications have, and will continue to, exercise the CPU more heavily and other smartphone components also contribute to mobile device power consumption, thermal constraints will be a major challenge to overcome into the foreseeable future.

Like any embedded processor, mobile CPUs operate under a strict set of power constraints. The absence of active cooling mechanisms forces CPUs to operate within a strict thermal power budget. Most mobile devices budget anywhere between 2.5 W to 5 W for their components to consume [16] and the SoC only gets a portion of it. Conventional wisdom is that mobile CPUs are designed to operate within a power budget of 1 W [17]. However, our data shows that recent generations appear to target an even higher budget of 1.5 W (Fig. 4). The latest five mobile CPUs that we studied reached about 1.5 W.

Mobile CPU power consumption can significantly exceed the SoC TDP independent of the other processing subsystems such as the GPU. Fig. 11a shows the manufacturer-*estimated* peak power consumption for SoC components, which we recovered from the S5S and S6 system configuration files that are available on the filesystem. Combining the power consumption of the A7 and A15 quad-core CPU clusters from



(a) Power budget breakdown.

(b) CPU versus other component power.

(c) Battery vs. screen size.

**Fig. 11: (a) Manufacturer reported power budgets. (b) CPU measured peak power has increased significantly compared to the other key mobile components. Multicore CPU power alone can exceed SoC-level TDP. (c) Battery capacity versus screen size (linear) relationship over time.**

either phone more than doubles the device's 3.5 W target TDP. The performance-oriented A15 cluster can achieve almost twice the target TDP by itself. Because the peak CPU power can significantly exceed the SoC TDP, the CPU is capable of self-inflicted thermal throttling. From our experience benchmarking these phones, today's modern mobile CPUs are already susceptible to thermal-induced performance throttling.

Combining the CPU's power requirements with other smartphone components' power requirements shows the severity of the problem from a holistic device-level perspective. Fig. 11b shows that the *measured* peak sustainable power of the mobile CPU has increased dramatically across the seven mobile device generations, and it is in fact worse than the manufacturer-estimated power shown in Fig. 11a. Starting from 2011, a single mobile core CPU is capable of consuming more power than the display and radio, thus closely approaching the system-level TDP limit. Including the display and radio implies the system operates at close to peak TDP. Running multiple cores only exacerbates the problem. On any of the last three mobile CPU generations, the multicore CPU alone can exceed the *entire mobile device's TDP* without including the radio and display units' power consumption.

**Energy-budget Limits** Unlike desktop CPUs, mobile devices are severely constrained by a battery-imposed energy budget. Battery technology has not experienced Moore's law-like improvements because of fundamental physics limitations [50]; the density of lithium-ion batteries has improved by only about 10% a year [18]. Therefore, the battery capacity of today's mobile devices is determined by the battery's volume, which is largely dictated by the device's screen size [19].

Fig. 11c compares the screen sizes and battery capacities of over 600 smartphones from 2006 to 2014. There is an almost linear correlation between the battery capacity and screen size. In the near future, screen size scaling may still be able to be the primary vehicle for increasing mobile device energy budgets. However, smartphone form factors are reaching form factor maturity. A recent study shows that about 98% of mobile users prefer a screen that is under five inches [20], which is roughly the size of Galaxy S4. Consequentially, we expect the total device energy budget to stay severely constrained.

However, mobile CPU energy efficiency improvements have plateaued, as we have demonstrated in Sec. 2.3. Given that users expect each mobile device generation to incorporate new sensors and other peripherals that also require energy from the same battery, it is clear that the mobile CPU, as a major energy consumer, needs to become more energy-efficient.

### 4.2. Limits of Desktop-like CPU Scaling

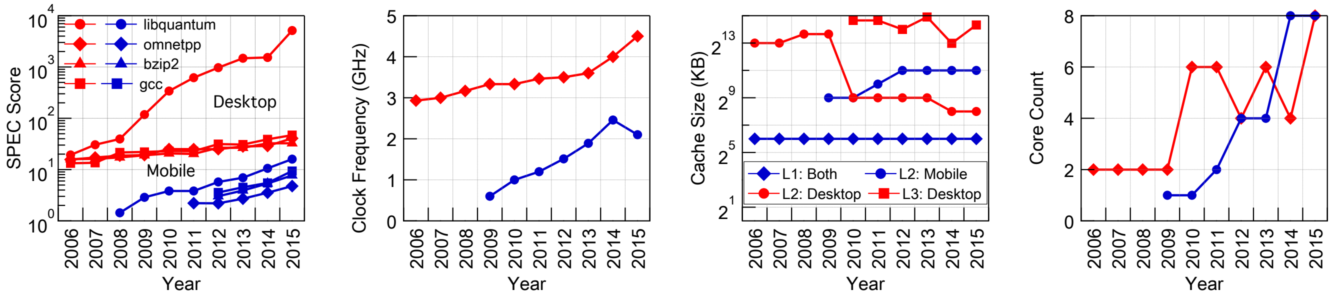
Over the past seven years, mobile CPUs incorporated over 20 years of desktop CPU design techniques to achieve performance and energy-efficiency at the expense of higher power consumption. Desktop CPUs have had decades to scale resources, ultimately succumbing to the "power wall." Therefore, desktop CPU trends provide us a reference point for how far conventional mobile CPU design techniques can be pursued.

Fig. 12 showcases the relationship between mobile and desktop CPUs across performance, clock frequencies, cache sizes, and multicore scaling. We mined the SPEC CPU 2006 database for Intel Core processors released between 2006 and 2015 [5] to representative mobile trends. The mobile CPU trends are based on the mobile CPUs we studied (Table 1).

Fig. 12a compares mobile CPU benchmark performance against desktop CPUs. Mobile CPU application performance *trends* have closely tracked desktop CPUs. The magnitudes of the trends between mobile and desktop are different largely due to the resource availabilities (i.e. transistors and area), rather than microarchitectural resource design. Each line represents the trend for a SPEC CPU 2006 benchmark across either the mobile or desktop CPUs. CPU generations (years) are on the *x*-axis, and SPEC scores are logarithmic on the y-axis. Even the performance of well-known outliers, such as `libquantum`, show a similar trend despite its significant memory level parallelism and locality characteristics [43].

Fig. 12b shows that mobile CPU frequencies are rapidly rising and closing the gap between desktop and mobile clock frequencies. Just as desktop CPU frequencies saturated as designs reach 100 W TDPs, it is likely that mobile CPU clock frequencies will saturate soon. Mobile CPUs have a power envelope orders of magnitude smaller (i.e. 3.5 - 5 W).

Fig. 12c shows that even cache sizes are maturing. The L1 cache sizes for both mobile and desktop CPUs have stayed



(a) SPECInt score comparison.

(b) Frequency comparison.

(c) Cache comparison.

(d) Core count comparison.

**Fig. 12: Comparison of desktop and mobile CPUs. Mobile CPUs are increasing like desktop CPUs, but there is still a huge performance gap.**

constant over time, but the L2 cache sizes that increased considerably over time in both mobile and desktop systems have reached a pinnacle point. Desktop CPU L2 cache sizes steadily increase through 2009 and suddenly drop to 512 KB with the introduction and growth of L3 caches. During the same period, the mobile CPU L2 cache sizes grew beyond desktop L2 caches towards 2 MB. While both the domains rely on scaling the last level cache for performance improvements, mobile CPU power and area budgets can constrain future growth.

Fig. 12d shows that desktop CPUs have adopted more cores over time with the hope that software parallelism will foster performance improvements. However, parallelizing single-threaded irregular programs is challenging, and parallel programming frameworks have not seen widespread adoption yet. Even if emerging parallel programming frameworks, such as Mare [13], RenderScript [14], and OpenCL [15] do catch on for mobile, power and energy constraints will likely to limit their effectiveness to achieve high performance computing.

## 5. The Mobile CPU in the Era of Specialization

Conventional desktop-like single- and multi-core scaling approaches to mobile CPU design are failing to keep pace with what mobile technology demands. As transistor densities exceed what a single-chip can fully power on at a given time (i.e. the dark silicon problem [28, 55]), specialization has become a promising technique to sustain the significant power, performance, and energy improvements that future computer systems necessitate. Today's mobile SoCs already consist of several specialized processing elements, and that number will continue to increase over time. A recent study showed a 3.5X increase in fixed-function accelerators across the six most recent Apple SoCs [51], and the ITRS anticipates thousands of different on-chip accelerators by 2022 [21].

The era of specialization will redefine the roles of how CPUs are used in future mobile systems. This section discusses three main roles that CPUs will play in future mobile systems and their design implications. Specifically, for many applications, the proliferation of specialized processing units will reduce CPU's compute responsibility while increasing its burden on managing the overall system complexity. Meanwhile, the CPU will continue to be a long-standing target for code portability and backwards compatibility for other applications.

**Irregularity Engine** Computing domains with an abundance of parallelism are the most promising candidates for acceleration, leaving the hard-to-parallelize regions for the CPU. For example, the heterogenous system architecture (HSA) framework, seeking adoption for programming future accelerator-rich SoCs, relies on a single-instruction multiple thread (SIMT) programming model that implicitly assumes that computations being accelerated are inherently parallel. However, most programs cannot be entirely executed on accelerators and will require some degree of CPU computation. The CPU will become an "irregular code accelerator."

The increasing amount of irregular code that future mobile CPUs need to handle indicate that many conventional CPU performance enhancement mechanisms will likely be less utilized than they would have been in the absence of accelerators. For example, a recent study on CPU-GPU computations [24], but applicable to CPUs involved in most modern heterogeneous computing paradigms, observed that instruction window size and stride-based memory prefetchers were less effective in CPU-GPU workloads than in conventional CPU workloads. This is because the ILP and locality that these mechanisms inherently rely on had been offloaded to the GPU. To extract what ILP and locality remain, future CPUs would benefit from better branch predictors, prefetching mechanisms, as prescribed in [24], as well as better cache management policies and other memory hierarchy optimizations.

**System Orchestrator** To date, operating system, runtime framework, and device driver code executes solely on CPUs. Under this paradigm, increasing the number of distinct processing elements within the SoC increases the system complexity that the CPU has to manage at runtime. Therefore, as more CPU computations are offloaded to accelerators more of the CPU's execution time will be devoted to system orchestration tasks, such as compute scheduling, resource configuration, data movement, and system monitoring.

System management tasks are typically sporadic and unpredictable, but lie on the program's critical path. These characteristics pose conflicting design requirements for future mobile CPUs. Today's mobile CPUs exploit prolonged periods of idleness for power efficiency optimizations. However, most of these optimizations come at the expense of responsiveness. For example, disabling CPU subsystems, such as the LLCs, can save power in idle CPUs, but subsequent CPU computations incur the performance penality to recover any state that was lost. A promising approach is to make CPUs aware of, in order to adapt to, accelerators' execution characteristics. Such an idea has been demonstrated with CPU-GPU computations [40] for DVFS, but can be extended to include other accelerators and be incorporated into prediction mechanisms for other CPU subsystems, such as memory prefetching.

**Legacy Code Target** Despite trends towards more hardware specialization in future mobile SoCs, many important applications have, and will continue to be, developed to primarily execute on the CPU. CPUs provide the most common and approachable mobile systems programming interface amongst the other processing technologies that exist today. The outstanding majority of developers write their programs in general-purpose programming languages that target the CPU and all SoCs posses at least one CPU subsystem. Therefore, CPUs play a key role in providing backwards compatability and code portability across mobile platforms.

Given the fact that CPUs will be responsible for executing both highly irregular programs and more conventional, regular programs, future CPUs should continue to embrace CPU-level heterogeneity trends present in today's CPUs. CPU-level heterogeneity provides a means for CPU designs optimized for programs that exhibit these two fundamentally different execution characteristics. Current heterogeneous CPUs, such as ARM's big.LITTLE technology [22], provide different CPUs core designs (i.e. out-of-order and in-order) to enable power-performance trade-offs. Future mobile CPU designs can extend this paradigm to optimize for well-established and frequently executed CPU workloads, such as the Android software stack [55] and Web browser [60].

## 6. Prior Work

Our study provides insight into how interactions between user experience, mobile applications, architecture and mobile device form factors shape and impact the mobile CPU design.

**Trend-based CPU Studies** Trend-based studies, specifically using real systems, help identify impactful research opportunities. Looking back on power and performance trends help identify impending bottlenecks and issues that may otherwise go unnoticed until it is too late. Recently, measurement-based trend studies were used to discuss ISAs [26] and desktop CPUs and managed languages [29]. Other trend-based studies use analytical models to identify the limits of clock [23], multicore [28] and memory bandwidth [48] scaling.

**User Experience Studies** Conventional user experience research consists of in-person user studies [47, 52–54, 56, 58, 59], where experiments are conducted in person, which limits the reach and diversity across participants. The majority of past user experience performance modeling research is geared towards producing power- and energy-efficiency techniques.

Our crowdsourcing framework allows us to include several orders of magnitude more participants spread across the world. Our work also bridges the gap between CPU design trends and user satisfaction by taking the feedback of over 25,000 users by proposing and using a novel crowdsourcing approach.

**CPU Evaluation Metrics** There are no shortage of evaluation metrics for CPU designs. However, these metrics largely ignore the end user. In particular, traditional hardware-centric perspectives such as performance-per-Watt, EDP [34], $ED^2P$ [41], ILP and TLP [25, 30, 32] only evaluate systems from a hardware efficiency perspective. While insightful, these metrics are not directly correlated with the end-to-end user-satisfaction that is important in mobile systems.

We take a different approach of using measured user satisfaction to explicitly bridge the gap between CPU performance capabilities and end-user satisfaction. The crowdsourcing based feedback allows us to quantitatively determine the extent to which a given CPU configuration achieves user satisfaction.

**Mobile Application Benchmarking** Mobile application benchmarking and characterization has recently become an active research area. Similar to our user study, almost all benchmarking efforts involve evaluating mainstream Android applications on ARM-based mobile processors. These prior studies are typically concerned with either architecture- [32] or microarchitectural-level [35, 37, 44] in the context of power and performance on a single architecture.

## 7. Conclusion

Over the past decade, mobile CPU designs have evolved to provide satisfactory end-user experiences by aggressively adopting desktop-like design techniques. However, as future mobile software evolves to become more complex and demands higher computational intensity, current mobile CPU design techniques cannot deliver the expected performance requirement under tight energy and thermal constraints. Our paper conveys the need for future mobile hardware designers to rethink mobile CPU design techniques and prepare for its role in SoC throughout the era of specialization.

## Acknowledgements

We would like to thank all of the people who have provided comments and suggestions. Specifically, we would like to thank fellow Trinity lab members Jingwen Leng, Yazhou Zu, Daniel Richens, and Zach Tschirhart, as well as Yakun Sophia Shao, Svilen Kanev, David Brooks, Niranjan Soundararajan, and Rami Al Sheikh. This research is supported in part by NSF awards CCF-1528045, CCF-1255892 and SRC 2013-HJ-2408, along with gifts from Google, Samsung and Intel.

## References

[1] Intel watches ARM as low-powered computing thrives. [Online]. Available: http://www.computerweekly.com/news/2240226532/Arm-is-a-competitor-we-take-very-seriously-says-Intel

[2] CoreMark Benchmarking for ARM Cortex Processors. [Online]. Available: http://infocenter.arm.com/help/topic/com.arm.doc.dai0350a/DAI0350A_coremark_benchmarking.pdf

[3] Geekbench. [Online]. Available: http://www.primatelabs.com/geekbench/

[4] SunSpider JavaScript Benchmark. [Online]. Available: https://www.webkit.org/perf/sunspider/sunspider.html

[5] Standard Performance Evaluation Corporation (SPEC). SPEC CPU2006 results. [Online]. Available: http://www.spec.org/cpu2006/results

[6] Cortex-A9 Reference Manual. [Online]. Available: http://infocenter.arm.com/help/topic/com.arm.doc.ddi0388f/DDI0388F_cortex_a9_r2p2_trm.pdf

[7] Cortex-A15 Reference Manual. [Online]. Available: http://infocenter.arm.com/help/topic/com.arm.doc.ddi0438c/DDI0438C_cortex_a15_r2p0_trm.pdf

[8] Cortex-A57 Reference Manual. [Online]. Available: http://infocenter.arm.com/help/topic/com.arm.doc.ddi0488c/DDI0488C_cortex_a57_mpcore_r1p0_trm.pdf

[9] Monsoon Power Monitor. [Online]. Available: https://www.msoon.com/LabEquipment/PowerMonitor/

[10] Amazon Mechanical Turk. [Online]. Available: https://www.mturk.com/mturk/welcome

[11] CompuBench. [Online]. Available: http://compubench.com/info.jsp

[12] Unreal Engine. [Online]. Available: https://www.unrealengine.com/

[13] Parallel Computing (MARE). [Online]. Available: https://developer.qualcomm.com/mobile-development/maximize-hardware/parallel-computing-mare

[14] RenderScript. [Online]. Available: http://developer.android.com/guide/topics/renderscript/compute.html

[15] OpenCL. [Online]. Available: https://www.khronos.org/opencl/

[16] The future of mobile CPUs, part 1: Today's fork in the road. [Online]. Available: http://arstechnica.com/gadgets/2013/02/the-future-of-mobile-cpus-part-1-todays-fork-in-the-road/

[17] Future of mobile CPUs, part 2: What's ahead for the major players? [Online]. Available: http://arstechnica.com/gadgets/2013/02/future-of-mobile-cpus-part-2-whats-ahead-for-the-major-players/2/

[18] Battery Statistics. [Online]. Available: goo.gl/3q9fUX

[19] Are smartphones getting larger because they have to? [Online]. Available: http://goo.gl/y5JYXf

[20] Size Matters for Connected Devices. Phablets Don't. [Online]. Available: http://goo.gl/I1LtQB

[21] System Drivers, ITRS, 2013. [Online]. Available: http://www.itrs.net/ITRS%201999-2014%20Mtgs,%20Presentations%20&%20Links/2013ITRS/2013Chapters/2013SysDrivers_Summary.pdf

[22] Big.LITTLE Technology. [Online]. Available: https://www.arm.com/products/processors/technologies/biglittleprocessing.php

[23] V. Agarwal, M. Hrishikesh, S. W. Keckler, and D. Burger, *Clock rate versus IPC: The end of the road for conventional microarchitectures*. ACM, 2000, vol. 28, no. 2.

[24] M. Arora, S. Nath, S. Mazumdar, S. B. Baden, and D. M. Tullsen, "Redefining the role of the cpu in the era of cpu-gpu integration," *Micro, IEEE*, 2012.

[25] G. Blake, R. G. Dreslinski, T. Mudge, and K. Flautner, "Evolution of thread-level parallelism in desktop applications," in *ACM SIGARCH Computer Architecture News*, vol. 38, no. 3. ACM, 2010, pp. 302–313.

[26] E. Blem, J. Menon, and K. Sankaralingam, "Power struggles: Revisiting the risc vs. cisc debate on contemporary arm and x86 architectures," in *High Performance Computer Architecture (HPCA2013), 2013 IEEE 19th International Symposium on*. IEEE, 2013, pp. 1–12.

[27] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE, 2009, pp. 248–255.

[28] H. Esmaeilzadeh, E. Blem, R. St Amant, K. Sankaralingam, and D. Burger, "Dark silicon and the end of multicore scaling," in *Prof. of ISCA*. IEEE, 2011, pp. 365–376.

[29] H. Esmaeilzadeh, T. Cao, Y. Xi, S. M. Blackburn, and K. S. McKinley, "Looking back on the language and hardware revolutions: measured power, performance, and scaling," in *ACM SIGARCH Computer Architecture News*, vol. 39, no. 1. ACM, 2011, pp. 319–332.

[30] K. Flautner, R. Uhlig, S. Reinhardt, and T. Mudge, "Thread-level parallelism and interactive performance of desktop applications," *ACM SIGOPS Operating Systems Review*, vol. 34, no. 5, pp. 129–138, 2000.

[31] C. Gao, A. Gutierrez, M. Ranaj, R. Dreslinski, T. Mudge, and C.-J. Wu, "A study of mobile device utilization," in *Proc. of ISPASS*, 2015.

[32] C. Gao, A. Gutierrez, R. G. Dreslinski, T. Mudge, K. Flautner, and G. Blakey, "A study of thread level parallelism on mobile devices," in *Proc. of ISPASS*, 2014.

[33] L. Gomez, I. Neamtiu, T. Azim, and T. Millstein, "Reran: Timing-and touch-sensitive record and replay for android," in *Proc. of ICSE*, 2013.

[34] R. Gonzalez and M. Horowitz, "Energy dissipation in general purpose microprocessors," *Solid-State Circuits, IEEE Journal of*, vol. 31, no. 9, pp. 1277–1284, 1996.

[35] A. Gutierrez, R. Dreslinski, A. Saidi, C. Emmons, N. Paver, T. Wenisch, and T. Mudge, "Full-system analysis and characterization of interactive smartphone applications," in *Proc. of IISWC*, 2011.

[36] R. Hariharan, Personal (email) communication, Apr. 18 2012, AMD.

[37] Y. Huang, Z. Zha, M. Chen, and L. Zhang, "Moby: A mobile benchmark suite for architectural simulators."

[38] L. Kishonti, Personal (email) communication, Jul. 24 2014, CompuBench.

[39] S. Komarov, K. Reinecke, and K. Z. Gajos, "Crowdsourcing performance evaluations of user interfaces," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2013, pp. 207–216.

[40] K. Ma, X. Li, W. Chen, C. Zhang, and X. Wang, "Greengpu: A holistic approach to energy efficiency in gpu-cpu heterogeneous architectures," in *Proc. of ICPP*. IEEE, 2014.

[41] A. J. Martin, M. Nyström, and P. I. Pénzes, "Et2: A metric for time and energy efficiency of computation," in *Power aware computing*. Springer, 2002, pp. 293–315.

[42] R. B. Miller, "Response time in man-computer conversational transactions," in *Proceedings of the December 9-11, 1968, Fall Joint Computer Conference, Part I*, ser. AFIPS '68 (Fall, part I). New York, NY, USA: ACM, 1968, pp. 267–277.

[43] O. Mutlu and T. Moscibroda, "Stall-time fair memory access scheduling for chip multiprocessors," in *Proceedings of the 40th Annual IEEE/ACM International Symposium on Microarchitecture*, 2007.

[44] D. Pandiyan, S.-Y. Lee, and C.-J. Wu, "Performance, energy characterizations and architectural implications of an emerging mobile platform benchmark suite-mobilebench," in *IISWC*, 2013.

[45] Y. Patt, "Requirements, bottlenecks, and good fortune: Agents for microprocessor evolution," in *Proceedings of IEEE*, 2001.

[46] R. Peri, Personal (email) communication, Nov. 23 2014, Intel.

[47] C. J. Reynolds, "The sensing and measurement of frustration with computers," Ph.D. dissertation, Massachusetts Institute of Technology, 2001.

[48] B. M. Rogers, A. Krishna, G. B. Bell, K. Vu, X. Jiang, and Y. Solihin, "Scaling the bandwidth wall: challenges in and avenues for cmp scaling," in *ACM SIGARCH Computer Architecture News*, vol. 37, no. 3. ACM, 2009, pp. 371–382.

[49] Samsung, Personal (email) communication, Apr. 28 2014.

[50] F. Schlachter, "No moore's law for batteries," in *Proc. of National Academy of Science of the United States of America*, 2013.

[51] Y. S. Shao, B. Reagen, G.-Y. Wei, and D. Brooks, "The aladdin approach to accelerator design and modeling," 2015.

[52] A. Shye, B. Ozisikyilmaz, A. Mallik, G. Memik, P. A. Dinda, R. P. Dick, and A. N. Choudhary, "Learning and leveraging the relationship between architecture-level measurements and individual user satisfaction," in *Proc. of ISCA*. Washington, DC, USA: IEEE Computer Society, 2008, pp. 427–438.

[53] A. Shye, Y. Pan, B. Scholbrock, J. S. Miller, G. Memik, P. A. Dinda, and R. P. Dick, "Power to the people: Leveraging human physiological traits to control microprocessor frequency," in *Microarchitecture, 2008. MICRO-41. 2008 41st IEEE/ACM International Symposium on*. IEEE, 2008, pp. 188–199.

[54] A. Shye, B. Scholbrock, and G. Memik, "Into the wild: studying real user activity patterns to guide power optimizations for mobile architectures," in *MICRO*, 2009.

[55] G. Venkatesh, J. Sampson, N. Goulding, S. Garcia, V. Bryksin, and M. B. Taylor, "Conservation cores: Reducing the energy of mature computations," in *ACM SIGARCH Computer Architecture News*. ACM, 2010.

[56] Z. Wang, F. X. Lin, L. Zhong, and M. Chishtie, "How far can client-only solutions go for mobile browser speed?" in *Proc. of WWW*, 2012.

[57] P. Yedlapalli, N. C. Nachiappan, N. Soundararajan, A. Sivasubramaniam, M. T. Kandemir, and C. R. Das, "Short-circuiting memory traffic in handheld platforms," in *Proceedings of the 47th Annual IEEE/ACM International Symposium on Microarchitecture*, 2014.

[58] Z. Zhao, M. Zhou, and X. Shen, "Satscore: Uncovering and avoiding a principled pitfall in responsiveness measurements of app launches," in *Proc. of Ubicomp*, 2014.

[59] Y. Zhu, M. Halpern, and V. J. Reddi, "Event-based scheduling for energy-efficient qos (eqos) in mobile web applications," in *Proc. of HPCA*, 2015.

[60] Y. Zhu and V. J. Reddi, "Webcore: architectural support for mobile web browsing," in *Proc. of the ISCA*, 2014.