The Report committee for Matthew Franklin Halpern
Certifies that this is the approved version of the following report:

Bridging the Gap Between Mobile CPU Design and User Satisfaction Via Crowdsourcing

APPROVED BY

SUPERVISING COMMITTEE:

Supervisor:

Vijay Janapa Reddi

Mohit Tiwari

Bridging the Gap Between Mobile CPU Design and User Satisfaction Via Crowdsourcing

by

Matthew Franklin Halpern, B.S.E.E.

REPORT

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE IN ENGINEERING

THE UNIVERSITY OF TEXAS AT AUSTIN

May 2016

Bridging the Gap Between Mobile CPU Design and User Satisfaction Via Crowdsourcing

Matthew Franklin Halpern, M.S.E.
The University of Texas at Austin, 2016

Supervisor: Vijay Janapa Reddi

This report aims to provide an understanding of how the mobile CPU designs have evolved and its influence on end-user satisfaction. To that end, a quantitative performance analysis is conducted across ten cutting-edge mobile CPU designs studied within top-selling off-the-shelf smartphones released over the past seven years. This analysis is then used to guide a large-scale user study spanning over 25,000 participants via crowdsourcing on the Amazon Mechanical Turk service. The user study asks participants to assess the responsiveness of interactive application use cases for a set of current-generation applications (e.g. Angry Birds and FaceBook) and next-generation applications (i.e. face recognition and augmented reality) relative to the performance capabilities of the devices studied. This framework allows us to quantitatively link how the mobile CPU designs studied impacted end-user satisfaction.

The study results indicate that mobile CPU designs have exhibited significant performance improvements through aggressive core scaling techniques prevalent in desktop CPUs. Just as was observed in desktop CPU design, these same techniques have lead to excessive mobile CPU power consumption. However, from

an end-user perspective this power consumption was not without success. Mobile CPUs have evolved to provide satisfactory experiences for the studied current-generation applications. The reason is that many of these applications rely heavily on single-threaded performance. Other, more recent applications, actually multi-thread user-critical parts of the applications, which also demonstrates that multi-core mobile CPUs are an important design consideration – contrary to conventional wisdom. However, looking ahead, the same mobile CPUs where not able to provide satisfactory experiences for many of the next-generation applications studied, questioning the sustainability of these power-hungry design techniques in future mobile CPU designs.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

The success of any mobile computing system is dicted by its end-users. As a result, mobile hardware design is driven by ambitious user requirements. End-users have come to expect that each mobile device generations heralds performance improvements while providing longer battery lifes and slimmer device form factors.

At the forefront of this hardware innovation is the mobile CPU. The mobile CPU has been, and will continue to be, a critical SoC component for three reasons. First, mobile applications are developed in general-purpose languages that target the CPU, such as Objective-C, Java, and Swift. Second, the most of the other mobile device components are orchestrated through the CPU as the OS kernel and device driver code executes on it. Lastly, accelerators are typically extracted from computational kernels previously executed on the CPU.

Mobile CPUs have evolved into high-performance processors in hopes of maximizing end-user satisfaction. However, 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 academia. Current mobile CPU architecture research exclusively focuses on the interactions between the hardware and software. This "system efficiency" approach largely ignores the end-user, treating them as a secondary concern at best. Yet end-users will ultimately determine whether or not these designs are successful.

This report seeks to provide an understanding of the mobile CPU's evolution by observing real-world mobile CPU trends in conjunction with end-user experience. It does so by measuring and quantifying 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 cutting-edge ARM-based mobile CPU technology. These mobile CPUs span nine different microarchitectures, six different process nodes and also include recent trends towards asymmetric multiprocessing and core customization.

To extend the conventional research scope to include the end-user and consider how generation mobile CPU design trends have affected them, we construct and conduct a novel crowdsourcing-based user study. Using Amazon's Mechanical Turk service, we are able to solicit over 25,000 participants – orders of magnitude more than prior work in computer architecture studies [50, 53–55]. 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 measurements show that mobile CPUs have evolved rapidly to deliver peak performance increases for better application responsiveness. Mobile CPU designs changed dramatically from 2008 to 2015, leveraging over 20 years of desktop CPU design techniques (i.e. resource scaling). However, as with desktop CPU designs, this aggressive resource scaling has resulted in excessive power consumption over time. Additionally, energy efficiency improvements have been stagnating as the power consumption increases have outpaced performance improvements.

Despite their power hungry nature, the results from the crowdsourced user study demonstrate that these mobile CPU design improvements have not been in vain. The participants' satisfaction improved for all of the current generation application use cases (e.g., YouTube, Photoshop). Therefore, the transition to out-of-order cores, and more recently big, aggressive cores coupled with little, efficient cores, has ultimately benefited end users. Overall, the results suggest that current mobile CPUs designs are overprovisioned for the bursty behavior of today's applications. Even with applications that utilize the GPU and other accelerators, the participants were still sensitive to CPU performance.

Looking ahead, our users did not find the mobile CPU performance satisfactory for many of the next-generation applications (e.g., real-time face detection), which motivates the need to better understand how to provide the power-efficienct performance improvements they necessitate in future designs.

Chapter 2

Motivation: Mobile CPU Design Trends[1]

To understand how mobile CPU designs have progressed to deliver performance, this chapter presents a quantitative performance analysis across ten ARM-based mobile CPUs found in top-selling smarthphones released from 2009 to 2015. Each smartphone encapsulates the cutting-edge mobile CPU technology available within a particular year.

The results demonstrate that mobile CPU performance has increased dramatically over time – ultimately at the expense of excessive power consumption. These performance improvements are the result of desktop-like resource scaling: more aggressive microarchitectural mechanisms, clock frequency increases, and memory hierarchy growth (as well as multi-core scaling, which is not considered in this section). Energy efficiency at peak performance also improved significantly but has recently started to diminish because of excessive power consumption.

2.1 Mobile CPU Selection and Experimental Approach

This section introduces the mobile CPUs that will be studied throughout the remainder of this report. The CPUs were carefully chosen to reflect the mobile CPU design trends most prevalant in the real-world.

[1]This chapter includes research previously published in [36]

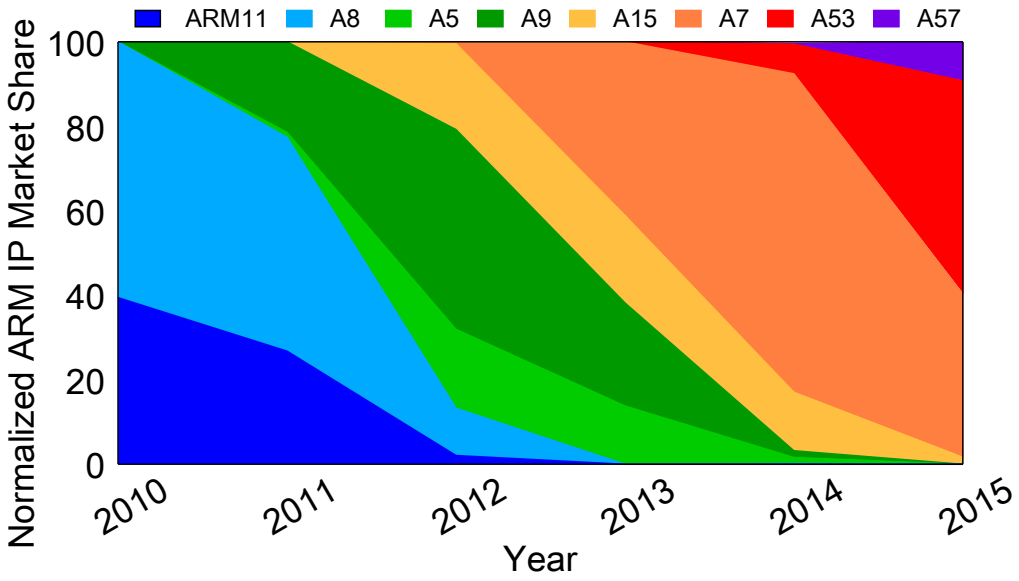


Fig. 2.1: Breakdown of Yearly ARM Cortex-A CPU Design Market Share.

2.1.1 Mobile CPU Selection Criteria

An important aspect of conducting any generational study is selecting the right "samples" to study. Our work focuses on ten ARM Cortex-A-based CPUs released between 2009 and 2015, which has been the dominant line of mobile CPUs used to date [1]. It is important to note that in this report "CPU" is used 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.

Mobile CPUs are being introduced at an unprecedented rate. To make an informed selection of what mobile CPUs to study, Fig. 2.1, was generated from data mined from over 1700 Android smartphone specifications. Fig. 2.1 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

Table 2.1: Mobile CPUs Under Study.

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

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.

The mobile CPUs studied in this report, shown in Table 2.1, track the mobile CPU design trends shown in Fig. 2.1. These tens mobile CPU designs comprehensively cover the past six years of mobile CPU design, covering the 32-bit ARM-based CPU microarchitectures, process technologies, and both symmetric and asymmetric multiprocessing trends prevalent throughout this time period. These are also the same CPU designs used in other mobile device form factors. For example, both the Samsung Galaxy Tab 12.6 tablet and Samsung Galaxy S5 (S5S) and Google Glass and Samsung Galaxy Nexus (N) utilize the same SoC families. From hereon forward and throughout the rest of the paper, we refer to each smartphone model by its abbreviation.

For completeness, different design methodologies between the various CPU manufacturers are also considered. Specifically, two CPUs vendor designs for each year from 2012 to 2014 are studied. 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.

2.1.2 Workloads

The goal of this section is to quantify performance, power, and energy efficiency across mobile CPU generations. To that end, we solely use industry standard CPU-intensive benchmarking applications to isolate the peak CPU performance capabilities within each system. We defer the study of interactive mobile applications under realistic use cases involving users to the next section.

The mobile CPUs are evaluated amongst well-established embedded and desktop system benchmarks; both within industry and the research community. EEMBC's `Coremark` benchmark is used to represent embedded benchmarking, which specifically targets the embedded market segment. The benchmark has been used in prior research to evaluate mobile CPUs [21], as well as in industry [2].

In addition, a subset of the SPEC CPU2006 integer benchmark suite [3] is used. Specifically, `gcc`, `libquantum`, `omnetpp`, `hmmer`, and `bzip2`, whose inclusion is limited due to compiler and storage issues. The `test` inputs are used due to memory limitations, also observed in [21]. While it may seem unconventional to use desktop CPU benchmarks to evaluate mobile CPUs, industry actually does this. Many industry companies have acknowledged the use of SPEC to evaluate future mobile CPU designs [37, 49, 52].

Sunspider [4], Geekbench [5], and Stream [44], have become popular benchmarking workloads amongst end-users and within technology journalism. Therefore, these "magazine workloads" are also used in the study.

2.1.3 Power Measurements

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 [21]. Battery-level power measurements are collected from each device using the Monsoon Power Meter [6], which has a five kilo-samples per second and performs self-calibration. Differential power measurements ($P_{active} - P_{idle}$) are used to isolate the CPU's dynamic power consumption. Extracting power consumption differentially removes static power consumption from idle components such as the display and the other (unused) SoC components. The radio and other components unrelated to our study are disabled throughout each power measurement.

2.1.4 Challenges, Precautions, and Assumptions

Physically extracting generational trends from representative mobile CPUs is challening. Ideally, such a study should strictly isolate differences between CPUs – holding operating system, memory (RAM) and other external factors constant across experiments. However, mobile processors are tightly integrated and cannot be easily isolated (as in desktop systems), and such a completely isolated study is infeasible. As a result, the main challenge is to extract CPU trends that are largely independent of the other system-level differences across the smartphones under study. These experiments reflect a best effort attempt to reduce the impact of these differences and only the conclusions drawn from them are meant to be robust to these external factors, following similar approaches as prior work [21, 29].

2.1.4.1 Operating System

Many of the smartphones utilize different OS versions from one another. These differences are minimized by the benchmark selection and compilation process. SPEC and CoreMark are designed to be robust to different kernel versions by limiting system calls throughout the program [3, 7]. In addition, to mitigate any dy-

namically linked library effects, the benchmarks are statically cross-compiled under the same toolchain so that each system executes the same binaries containing the exact same library code. The only exception is the S6, which uses ARMv8 binaries (but are still statically compiled).

2.1.4.2 Memory (DRAM)

Memory does not significantly affect our measurements and conclusions. Directly measuring the CPU and DRAM power rails on the ODroid-XU development platform (the same components in S4S) indicates that DRAM accounts for less than 9% of the total power across the benchmarks. Prior work observed similar proportions [23, 24]. The differential power measurements also filter out the DRAM's self-refresh and static power consumption. From a performance perspective, it is reasonable to assume that each DRAM module is carefully selected by the mobile device manufacturer to adequately feed data to its corresponding CPU.

2.2 Performance Analysis

Fig. 2.2 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

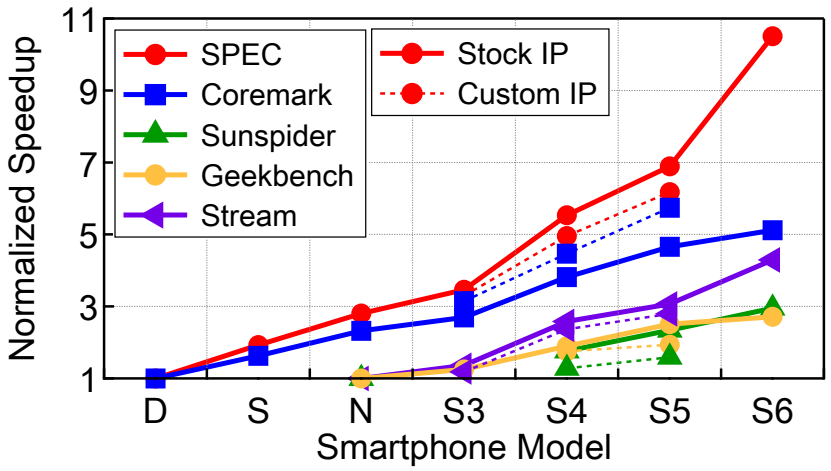


Fig. 2.2: Mobile CPU Single-core Performance Trends.

mobile CPU generations. As Table 2.1 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. 2.3 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 [8] 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 [8]. 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 [9]. The A57 (2014),

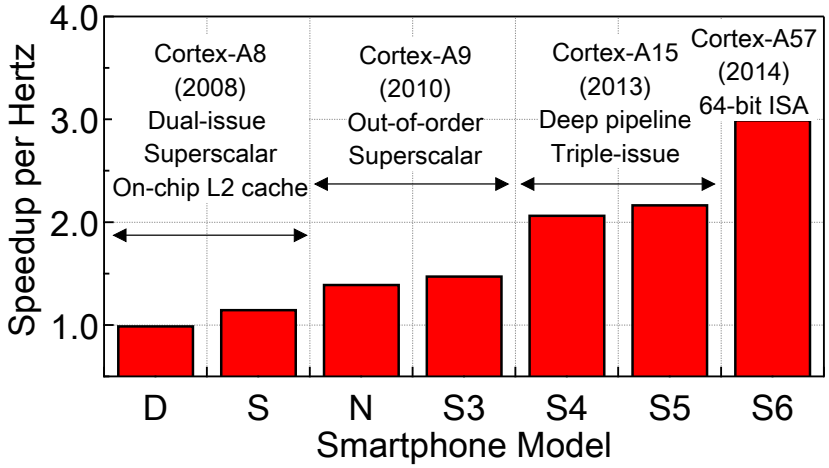


Fig. 2.3: Mobile CPU Instruction-Per-Cyle Trends.

used in the S6, incorporates a new 64-bit instruction set architecture (ISA) into an A15-like design [10].

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 [46] 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 Power and Energy Analysis

Fig. 2.4 shows the power consumption trend across mobile CPU generations. Initially, power consumption mostly reduced as performance improved from

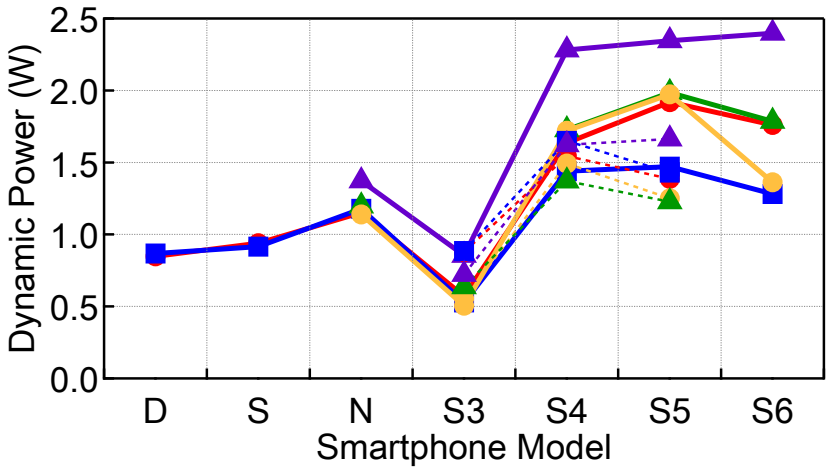


Fig. 2.4: Mobile CPU Single-core Power Consumption Trends.

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. 2.2 and Fig. 2.4 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

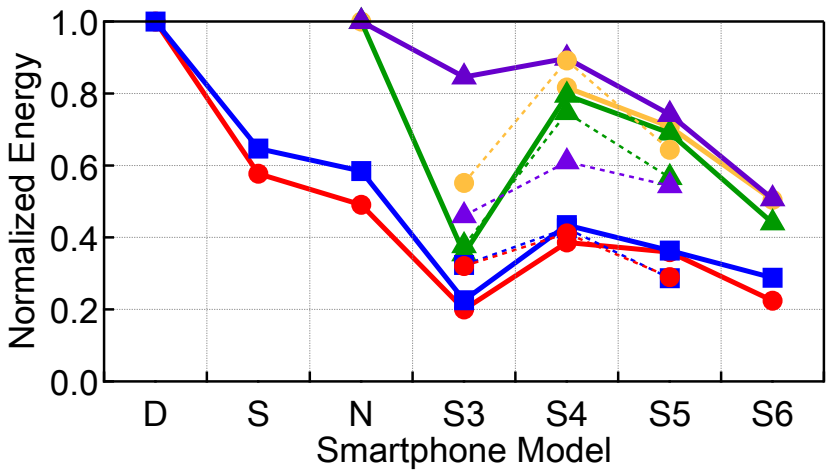


Fig. 2.5: Mobile CPU Single-core Energy Efficiency Trends.

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 [48]. Process innovations do not occur frequently, so we do not see large improvements (or dips) in CPU power consumption in the following generations.

Fig. 2.5 shows CPU energy consumption across the seven 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.

Chapter 3

Methodology: Crowdsourcing a User Study[1]

To determine whether the generational mobile CPU performance enhancements justify the power increases, we perform a novel crowdsourced user study that includes over 25,000 participants solicited through Amazons Mechanical Turk service [11]. Our large-scale user study allows us to comprehensively assess mobile users' sensitivities to different CPU architecture and performance configurations with high statistical confidence. Most prior work [55, 57, 59] only uses a handful of users, and thus it is unlikely that one can derive strong conclusions with high statistical confidence.

3.1 Interactive Application Selection

We study a broad range of popular Android applications, shown in Table 3.1. 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, 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.

[1]This chapter includes research previously published in [36]

Table 3.1: Interactive Application Used In User Study

	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

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 [12], an industry-strength benchmark suite, used by various mobile device manufacturers [40] that consists of user-facing application demos built on top of computationally intensive kernels.

We select a broad range of applications for our user study, which are shown in Table 3.1. These applications are not only amongst the most popular Android applications downloaded from Google Play [13], but more importantly, they represent a diverse set of usage and computational characteristics typical of today's interactive mobile applications. In addition, we also incorporate applications from emerging application domains into our study. These applications are contained within CompuBench, an industry-strength benchmark suite [12], used by various mobile device manufacturers [40], that consists of user-facing demos that rely on computationally intensive next-generation application domains, such as augmented reality, physics simulation and advance image processing.

Our application selection criteria decomposes applications beyond typical application domain categories into user- and hardware-level metrics, as shown in Table 3.1. Our user-oriented application selection criteria includes various user behaviors, i.e., waiting for a webpage to load versus watching a video. To convey the variety of interactiveness across applications, we present the number of interactive events that take place within each use case in the "User Events" column. Each interactive event corresponds to an individual tap, swipe, or drag activity.

In order to faithfully examine the mobile CPU design in our study, we also ensure select application whose use cases demonstrate diverse computational characteristics. In particular, we use thread-level parallelism (TLP) collected using the `systrace` Android utility. TLP quantifies the amount of parallelism within each application. We will discuss these characteristics and their impact on user satisfaction and mobile CPU design throughout the remainder of the section.

While we aim to broadly cover mainstream application domains, we intentionally include three different gaming applications that exhibit different usage and computational behaviors. A recent study observed that 32% of mobile device usage is devoted to gaming, with no one particular outstanding game type [14]. For similar interactive session durations (36-44 seconds), these three games exhibit different interactivity and computational characteristics. `Gladiator` is a first-person fighter where the user presses a button to swing a sword at an opponent, requiring 31 presses to win the level. `Angry Birds` required six interactions to navigate several menus and a single drag to complete the level, and `Epic Citadel` requires 15 taps to navigate through the game's environment.

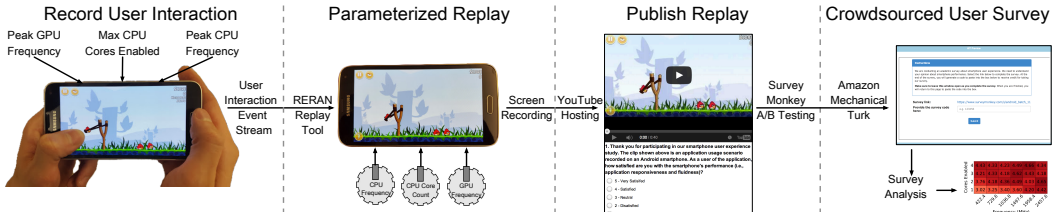


Fig. 3.1: Crowdsourced User Study Methodology.

3.2 Experimental Flow

Our crowdsourced study consists of having participants rank their satisfaction while we replay representative application use cases under various CPU performance configurations, i.e., core counts and clock frequency – a total of 24 configurations across 13 workloads. Our study presents a unique set of challenges and showcases a new approach for conducting studies at this scale in the future. Therefore, we present the rationale behind key choices and design decisions taken for the study.

3.2.1 Mechanical Turk

Amazon's Mechanical Turk [11] service is a marketplace for Human Intelligence Tasks (HITs) where requesters post tasks with a price for workers to perform. Crowdsourcing through Mechanical Turk is well-established in other research areas, such as for computer vision training data [26] and answering psychological questionnaires [22].

We solicited over 25,000 users for our study. We got high user engagement by posting $0.10 HITs for workers. In order to establish statistical confidence in our study, we each configuration is rated by at least 50 participants. Fig. 3.1 outlines the MTurk-based study methodology, which we describe next.

3.2.2 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` [32], which is a low-overhead, deterministic touchscreen event injection tool that was developed for the Android platform.

We record each application with the `S5Q` fixed to operating at peak performance (i.e., all four CPU cores at 2.4 GHz). We deem this the baseline user interaction trace. Intuitively, recording the trace at peak performance maximizes the likelihood of capturing a seamless user experience where the user does not feel constrained by mobile CPU performance [45].

3.2.3 Parameterized Replay and Phone Mapping

To investigate 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 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 are able to simulate the CPU performance configurations found across the earlier mobile CPU generations we study.

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 approxi-

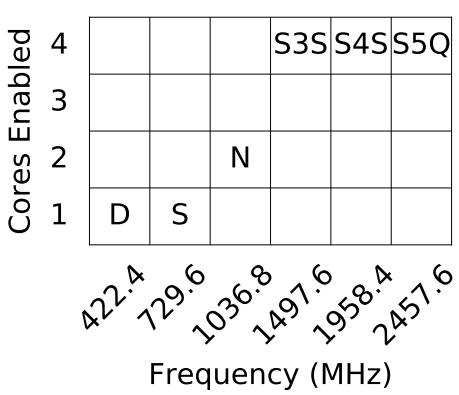


Fig. 3.2: S5Q CPU mapping.

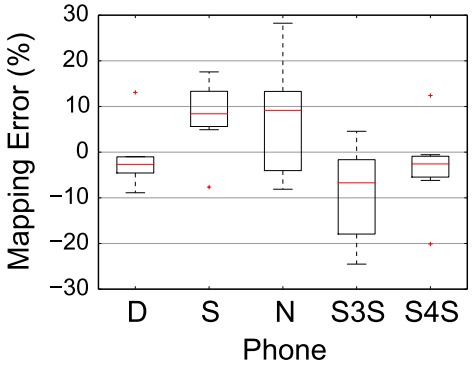


Fig. 3.3: Mapping error.

mated with CPU core count. Fig. 3.2 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. 3.3 shows that the mapping error is less than 10% for each phone.

3.2.4 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 the `youtube.com` website.

3.2.5 Crowdsourced User Study

The replayed user interaction video is embedded into a publicly accessible online survey at `surveymonkey.com`. Each user satisfaction survey consists of a single, randomly selected video clip and a single 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.

3.2.6 Precautions, Limitations and Assumptions

We are careful to minimize the impact of the survey question and answer choices on the participants' judgment. The survey establishes that the study is about the perceived mobile device performance, rather than the quality of the application itself, for the single video clip in question. We chose to not provide a reference video for comparison to prevent a bias in the results.

Real mobile device users interact with a physical device, whereas our survey results are based on participants watching a video. Therefore, there may be a slight mismatch in user satisfaction results. To develop an intuition about this effect, We evaluated the videos across a small group of users in-house. The crowdsourced results are consistent with the trends we observe in-person, which is demonstrated more generally in [41]. We did not rely on in-person user studies because they do not allow us to gain enough samples for statistical confidence in a scalable manner. In addition, having each participant watch a video clip, instead of performing direct interaction with the mobile device, still allows the participants' situational awareness to focus on the perceived system performance [25].

Chapter 4

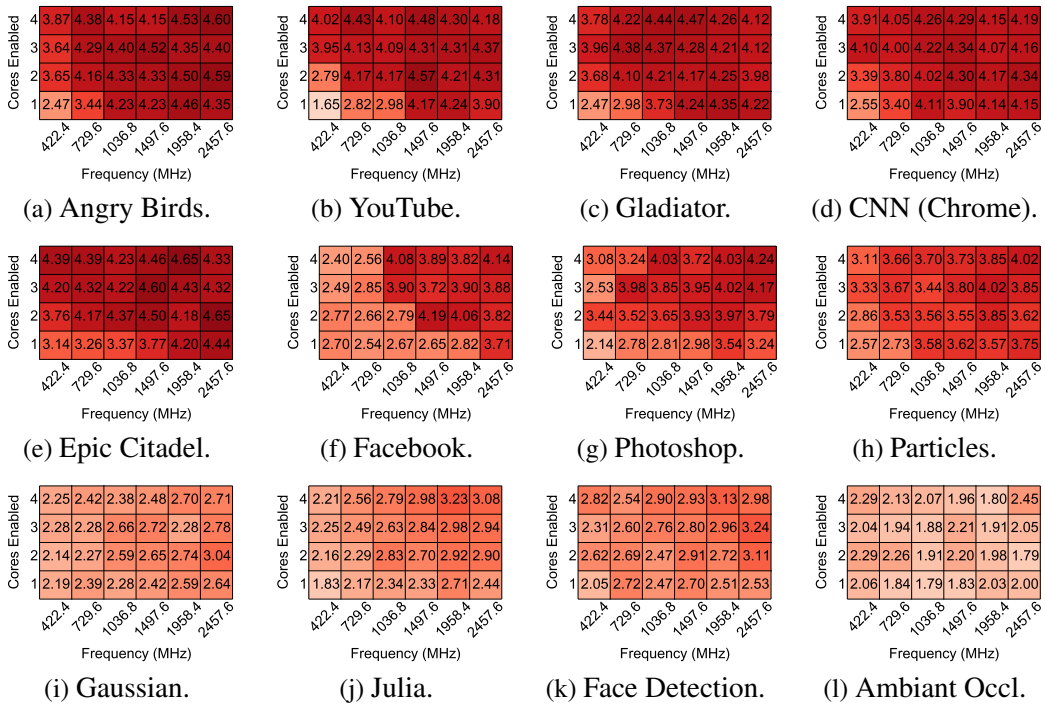
Findings: Architecting for End-Users[1]

4.1 Interpretting Results

We present the results of our crowdsourced user study for the workloads in Fig. 4.1. Each heatmap corresponds to an application in Table 3.1. The heatmap cells represent the user satisfaction score for a particular $(\text{single-core, multicore})$ 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. 4.1a 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$)$.

[1]This chapter includes research previously published in [36]



(a) Angry Birds. (b) YouTube. (c) Gladiator. (d) CNN (Chrome).

(e) Epic Citadel. (f) Facebook. (g) Photoshop. (h) Particles.

(i) Gaussian. (j) Julia. (k) Face Detection. (l) Ambiant Occl.

Fig. 4.1: User Satisfaction Results for Single- and Multi-core CPU Analysis

4.2 Role of Single-core CPU Performance Improvements

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. 4.1a) and Gladiator (Fig. 4.1c), and webpage loading (Fig. 4.1d), 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. 4.1b) each has a thread-level parallelism (TLP) [30] close to two (Table 3.1), 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. 4.1e) uses the computationally intensive Unreal Game Engine [15]. Single-core performance on par with an `S4S` A15 core can provide a satisfactory experience to participants.

4.3 Role of Multi-core CPU Performance Improvements

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. 4.1a) and Gladiator (Fig. 4.1c), and webpage loading (Fig. 4.1d), 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. 4.1b) each has a thread-level parallelism (TLP) [30] close to two (Table 3.1), 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. 4.1e) uses the computationally intensive Unreal Game Engine [15]. Single-core performance on par with an `S4S` A15 core can provide a satisfactory experience to participants.

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 multithread-

ing. Photoshop (Fig. 4.1g), 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. 4.1h), 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. 4.1f) 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 [59]. 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.

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 [16], RenderScript [17], and OpenCL [18], 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. 4.1i). `Julia` (Fig. 4.1j) 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.

4.4 Role of GPU and Other Accelerators' Performance

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 these processing elements also impact end-user satisfaction. Incorporating such an analysis for these other components also provides the means to compare their contributions against the CPU in terms of end-user satisfaction.

In addition to the CPU, the proposed crowdsourcing metholodogy can also be applied to other SoC components. To demonstrate the extensibility of this methodology to these other components, this report considers the role of the GPU performance. All mobile applications exercise the mobile GPU to some degree, making it the most heavily utilized SoC accelerator.

User satisfaction was not sensitive to the performance differences across the S5Q's Adreno 330 GPU for almost all of the applications studied. Fig. 4.2 shows user satisfaction as the S5Q's Adreno 330 GPU frequency is sweeped 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

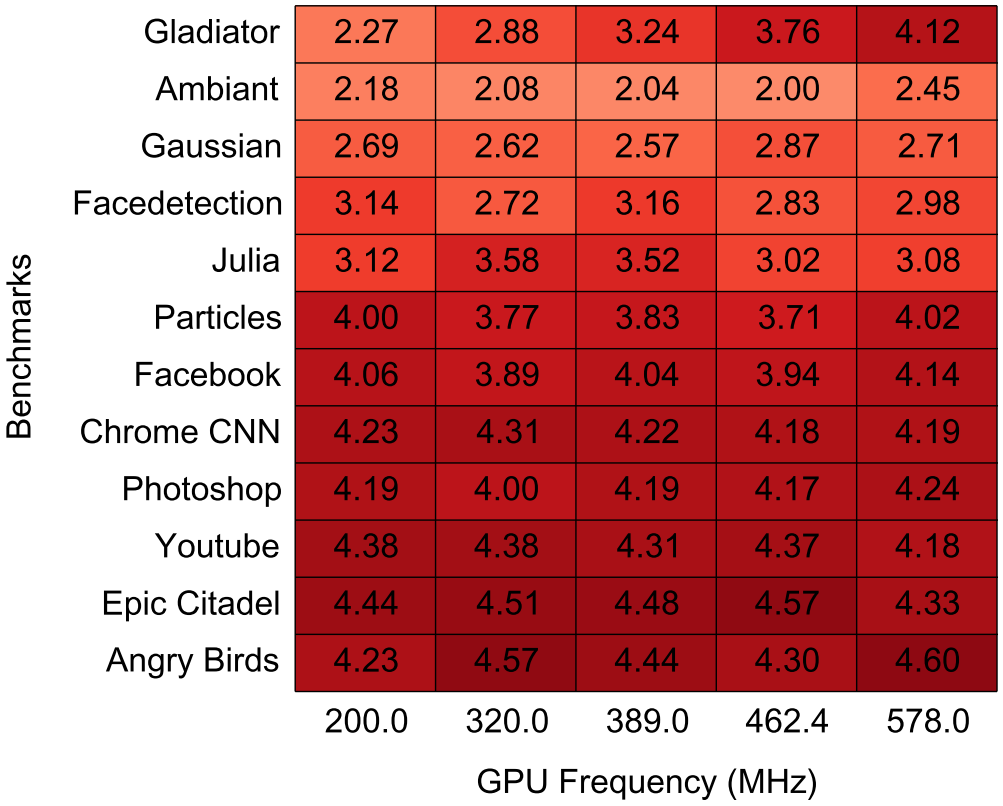


Fig. 4.2: User Satisfaction Results for GPU Analysis

of screen updates. Thus, user satisfaction increases with frequency by nearly four-fold 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 criti-

cal execution path even though computations are offloaded to these accelerators. Fig. 4.1b 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 [58].

Chapter 5

Related Work[1]

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

5.1 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 [21] and desktop CPUs and managed languages [29]. Other trend-based studies use analytical models to identify the limits of clock [19], multicore [28] and memory bandwidth [51] scaling.

5.2 User Experience Studies

Conventional user experience research consists of in-person user studies [34, 50, 53–55, 57, 59, 60], where experiments are conducted in person, which limits the reach and diversity across participants. The majority of past user experience perfor-

[1] This chapter includes research previously published in [36]

mance 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 20,000 users by proposing and using a novel crowdsourcing approach.

5.3 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 [33], ED^2P [43], ILP and TLP [20, 30,31] 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.

5.4 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- [31] or microarchitectural-level [35, 39, 46] in the context of power and performance on a

single architecture.

5.5 Crowdsourcing

Crowdsourcing [38] has been used for some time in a variety of research areas from machine-learning [26] to psychology [22] to astronomy [42] to biology [56]. Our work is most similar to HCI-related crowdsourcing [27], which uses the crowd to conduct user studies. The key distinction of our work is our emphasis on using the results to evaluate computer hardware mechanisms, as opposed to UI/UX design. Since this work begain, crowdsourcing has also begun to emerge in other computer systems research [47]. So far, these works focus on identifying the user-perceived quality of a program's final result (i.e. an image) as opposed to the user-perceived quality of interacting with the program (i.e. interactive applications) as was done in this work.

Chapter 6

Conclusion

This reports demonstrates the ability to incorporate the end-user into mobile computer architecture evaluation. In the face of power, thermal, and energy constraints, understanding what design decisions are going to impact the end-user will be an important aspect of future mobile CPU design. While the mobile CPU is amongst the most important components within the mobile device in terms of its affect on the end-user, there is a need to conduct these same kinds of analyses with other mobile hardware components both at the compute-level (e.g. GPU) and device-level (e.g. display and radio). The techniques and methodologies outlined in this report provide insight as to how to go about doing so.

Index

Bibliography

[1] Intel watches ARM as low-powered computing thrives.

[2] CoreMark Benchmarking for ARM Cortex Processors.

[3] Standard Performance Evaluation Corporation (SPEC). SPEC CPU2006 results.

[4] SunSpider JavaScript Benchmark.

[5] Geekbench.

[6] Monsoon Power Monitor.

[7] EEMBC CoreMark.

[8] Cortex-A9 Reference Manual.

[9] Cortex-A15 Reference Manual.

[10] Cortex-A57 Reference Manual.

[11] Amazon Mechanical Turk.

[12] CompuBench.

[13] Google Play.

[14] Apps Solidfy Leadership Six Years into the Mobile Revolution.

[15] Unreal Engine.

[16] Parallel Computing (MARE).

[17] RenderScript.

[18] OpenCL.

[19] Vikas Agarwal, MS Hrishikesh, Stephen W Keckler, and Doug Burger. *Clock rate versus IPC: The end of the road for conventional microarchitectures*, volume 28. ACM, 2000.

[20] Geoffrey Blake, Ronald G Dreslinski, Trevor Mudge, and Krisztián Flautner. Evolution of thread-level parallelism in desktop applications. In *ACM SIGARCH Computer Architecture News*, volume 38, pages 302–313. ACM, 2010.

[21] Emily Blem, Jaikrishnan Menon, and Karthikeyan 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*, pages 1–12. IEEE, 2013.

[22] Michael Buhrmester, Tracy Kwang, and Samuel D Gosling. Amazon's mechanical turk a new source of inexpensive, yet high-quality, data? *Perspectives on Psychological Science*, 6(1):3–5, 2011.

[23] Aaron Carroll and Gernot Heiser. An analysis of power consumption in a smartphone. In *USENIX annual technical conference*, pages 271–285, 2010.

[24] Aaron Carroll and Gernot Heiser. The systems hacker's guide to the galaxy: Energy usage in a modern smartphone. In *Proceedings of the 4th Asia-Pacific Workshop on Systems*, page 5. ACM, 2013.

[25] Andrew R Dattel, Jason E Vogt, Jessica K Fratzola, Daniel P Dever, Matthew Stefonetti, Chelsea C Sheehan, Marissa C Miller, and Joseph A Cavanagh. The gorillas role in relevant and irrelevant stimuli in situation awareness and driving hazard detection. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, volume 55, pages 924–928. Sage Publications, 2011.

[26] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009.

[27] Serge Egelman, Ed H Chi, and Steven Dow. Crowdsourcing in hci research. In *Ways of Knowing in HCI*, pages 267–289. Springer, 2014.

[28] Hadi Esmaeilzadeh, Emily Blem, Renee St Amant, Karthikeyan Sankaralingam, and Doug Burger. Dark silicon and the end of multicore scaling. In *Prof. of ISCA*, pages 365–376. IEEE, 2011.

[29] Hadi Esmaeilzadeh, Ting Cao, Yang Xi, Stephen M Blackburn, and Kathryn S McKinley. Looking back on the language and hardware revolutions: measured power, performance, and scaling. In *ACM SIGARCH Computer Architecture News*, volume 39, pages 319–332. ACM, 2011.

[30] Kristián Flautner, Rich Uhlig, Steve Reinhardt, and Trevor Mudge. Thread-level parallelism and interactive performance of desktop applications. *ACM SIGOPS Operating Systems Review*, 34(5):129–138, 2000.

[31] Cao Gao, Anthony Gutierrez, Ronald G. Dreslinski, Trevor Mudge, Krisztian Flautner, and Geoffery Blakey. A study of thread level parallelism on mobile

devices. In *Proc. of ISPASS*, 2014.

[32] Lorenzo Gomez, Iulian Neamtiu, Tanzirul Azim, and Todd Millstein. Reran: Timing-and touch-sensitive record and replay for android. In *Proc. of ICSE*, 2013.

[33] Ricardo Gonzalez and Mark Horowitz. Energy dissipation in general purpose microprocessors. *Solid-State Circuits, IEEE Journal of*, 31(9):1277–1284, 1996.

[34] Ashish Gupta, Bin Lin, and Peter A Dinda. Measuring and understanding user comfort with resource borrowing. In *High performance Distributed Computing, 2004. Proceedings. 13th IEEE International Symposium on*, pages 214–224. IEEE, 2004.

[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] Matthew Halpern, Yuhao Zhu, and Vijay Janapa Reddi. Mobile cpu's rise to power: Quantifying the impact of generational mobile cpu design trends on performance, energy, and user satisfaction. In *Proceedings of the 22nd International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 2016.

[37] Rema Hariharan. Personal (email) communication, Apr. 18 2012. AMD.

[38] Jeff Howe. The rise of crowdsourcing. *Wired magazine*, 14(6):1–4, 2006.

[39] Yongbing Huang, Zhongbin Zha, Mingyu Chen, and Lixin Zhang. Moby: A mobile benchmark suite for architectural simulators.

[40] Laszlo Kishonti. Personal (email) communication, Jul. 24 2014. CompuBench.

[41] Steven Komarov, Katharina Reinecke, and Krzysztof Z Gajos. Crowdsourcing performance evaluations of user interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 207–216. ACM, 2013.

[42] Chris J Lintott, Kevin Schawinski, Anže Slosar, Kate Land, Steven Bamford, Daniel Thomas, M Jordan Raddick, Robert C Nichol, Alex Szalay, Dan Andreescu, et al. Galaxy zoo: morphologies derived from visual inspection of galaxies from the sloan digital sky survey. *Monthly Notices of the Royal Astronomical Society*, 389(3):1179–1189, 2008.

[43] Alain J Martin, Mika Nyström, and Paul I Pénzes. Et2: A metric for time and energy efficiency of computation. In *Power aware computing*, pages 293–315. Springer, 2002.

[44] John McCalpin. Stream benchmark. *Link: www. cs. virginia. edu/stream/ref. html# what*, 1995.

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

[46] Dhinakaran Pandiyan, Shin-Ying Lee, and Carole-Jean Wu. Performance, energy characterizations and architectural implications of an emerging mobile platform benchmark suite-mobilebench. In *IISWC*, 2013.

[47] Jongse Park, Emmanuel Amaro, Divya Mahajan, Bradley Thwaites, and Hadi Esmaeilzadeh. Approxigame: Towards crowd-sourcing quality target determination in approximate computing. 2016.

[48] Yale Patt. Requirements, bottlenecks, and good fortune: Agents for microprocessor evolution. In *Proceedings of IEEE*, 2001.

[49] Ramesh Peri. Personal (email) communication, Nov. 23 2014. Intel.

[50] Carson Jonathan Reynolds. *The sensing and measurement of frustration with computers*. PhD thesis, Massachusetts Institute of Technology, 2001.

[51] Brian M Rogers, Anil Krishna, Gordon B Bell, Ken Vu, Xiaowei Jiang, and Yan Solihin. Scaling the bandwidth wall: challenges in and avenues for cmp scaling. In *ACM SIGARCH Computer Architecture News*, volume 37, pages 371–382. ACM, 2009.

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

[53] Alex Shye, Berkin Ozisikyilmaz, Arindam Mallik, Gokhan Memik, Peter A. Dinda, Robert P. Dick, and Alok N. Choudhary. Learning and leveraging the relationship between architecture-level measurements and individual user satisfaction. In *Proc. of ISCA*, pages 427–438, Washington, DC, USA, 2008. IEEE Computer Society.

[54] Alex Shye, Yan Pan, Benjamin Scholbrock, J Scott Miller, Gokhan Memik, Peter A Dinda, and Robert 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*, pages 188–199. IEEE, 2008.

[55] Alex Shye, Benjamin Scholbrock, and Gokhan Memik. Into the wild: studying real user activity patterns to guide power optimizations for mobile architectures. In *MICRO*, 2009.

[56] Brian L Sullivan, Christopher L Wood, Marshall J Iliff, Rick E Bonney, Daniel Fink, and Steve Kelling. ebird: A citizen-based bird observation network in the biological sciences. *Biological Conservation*, 142(10):2282–2292, 2009.

[57] Zhen Wang, Felix Xiaozhu Lin, Lin Zhong, and Mansoor Chishtie. How far can client-only solutions go for mobile browser speed? In *Proc. of WWW*, 2012.

[58] Praveen Yedlapalli, Nachiappan Chidambaram Nachiappan, Niranjan Soundararajan, Anand Sivasubramaniam, Mahmut T Kandemir, and Chita R Das. Short-circuiting memory traffic in handheld platforms. In *Proceedings of the 47th Annual IEEE/ACM International Symposium on Microarchitecture*, 2014.

[59] Zhijia Zhao, Mingzhou Zhou, and Xipeng Shen. Satscore: Uncovering and avoiding a principled pitfall in responsiveness measurements of app launches. In *Proc. of Ubicomp*, 2014.

[60] Yuhao Zhu, Matthew Halpern, and Vijay Janapa Reddi. Event-based scheduling for energy-efficient qos (eqos) in mobile web applications. In *Proc. of HPCA*, 2015.