Two Billion Devices and Counting

An Industry Perspective on the State of Mobile Computer Architecture

Vijay Janapa Reddi, Hongil Yoon, and Allan Knies
Google

Mobile computing has grown drastically over the past decade. Despite the rapid pace of advancements, mobile device understanding, benchmarking, and evaluation are still in their infancies, both in industry and academia. This article presents an industry perspective on the challenges facing mobile computer architecture, specifically involving mobile workloads, benchmarking, and experimental methodology, with the hope of fostering new research within the community to address pending problems. These challenges pose a threat to the systematic development of future mobile systems, which, if addressed, can elevate the entire mobile ecosystem to the next level.

Mobile devices have come a long way from the first portable cellular phone developed by Motorola in 1973. Most modern smartphones are good enough to replace desktop computers. A smartphone today has enough computing power to be on par with the fastest supercomputers from the 1990s. For instance, the Qualcomm Adreno 540 GPU found in the latest smartphones has a peak compute capability of more than 500 Gflops, putting it in competition with supercomputers that were on the TOP500 list in the early to mid-1990s.

Mobile computing has experienced an unparalleled level of growth over the past decade. At the time of this writing, there are more than 2 billion mobile devices in the world.[1] But perhaps even more importantly, mobile phones are showing no signs of slowing in uptake. In fact, smartphone adoption rates are on the rise. The number of devices is rising as mobile device penetration increases in markets like India and China. It is anticipated that the number of mobile subscribers will grow past 6 billion in the coming years.[2] As Figure 1 shows, while the Western European and North American markets are reaching saturation, the vast majority of growth is coming from countries in Asia. Given that only 35 percent of the world's population has thus far adopted mobile technology, there is still significant room for growth and innovation.

Published by the IEEE Computer Society
0272-1732/18/$33.00 ©2018 IEEE

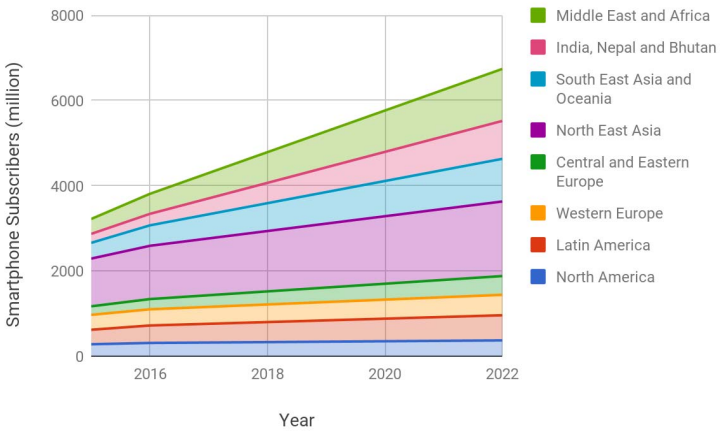

Figure 1. Mobile technology is still a strongly growing market. Only 35 percent of the world has so far adopted mobile technology.

The two strong drivers behind the widespread and rapid adoption of mobile devices worldwide are the ARM processors and Android operating system (OS). Licensing ARM's IP lowers the barrier of entry into the smartphone market. At the same time, the open-source Android OS allows the vendors to customize the OS to their specific hardware and provide unique user interfaces. A vast number of , if not all, smartphones today ship with ARM processors, and more than 80 percent of all smartphones in the world run using the Android OS.

In this article, we focus on the hardware. ARM processors are typically customized and packaged into different system-on-chip (SoC) architectures, or chipsets, that enable hardware vendors to differentiate themselves from one another by providing unique performance, power, and functional capabilities. As a result, as Figure 2 shows, the total number of ARM processor shipments has steadily increased over time. ARM shipments vastly outpace x86 processor shipments, where the ability to customize the processor is non-existent outside the walls of Intel and AMD.

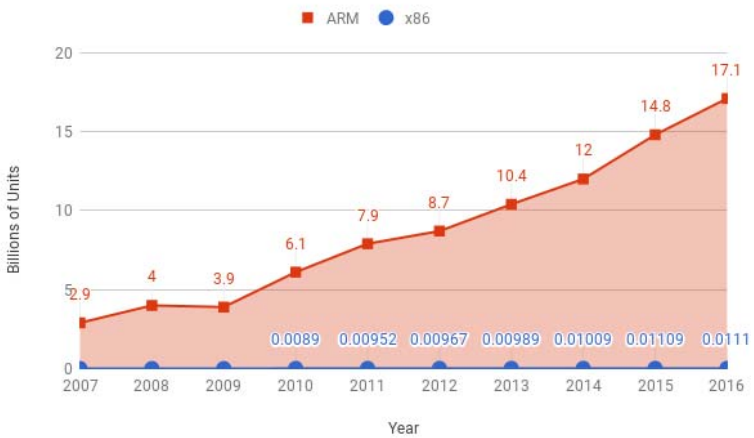

Figure 2. The number of x86 and ARM shipments from 2007 to 2016.[3,4] ARM processor shipments far outpace the number of x86 units shipped each year.

Processor customization, as a result of consumers' diverse demands, has led to a diverse marketplace, filled with low- to high-end mobile computing chipsets. As Figure 3 shows, during the period between 2011 and 2015, the number of SoC architectures that existed in the market each year (as adopted by smartphones and other mobile devices such as tablets) increased steadily.

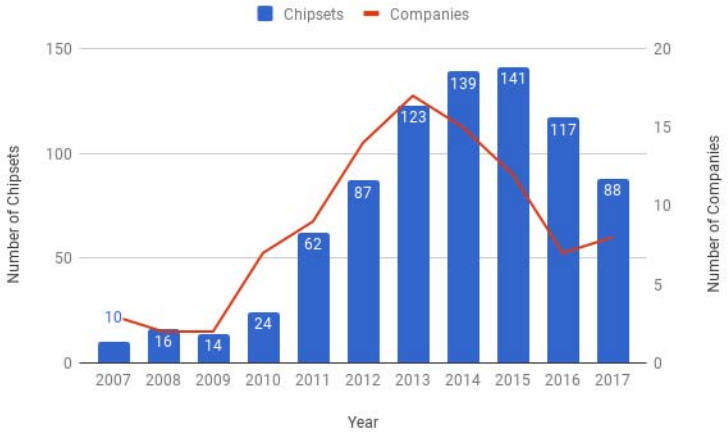

Figure 3: SoC diversification from 2007 to 2017. We mined and analyzed data from gsmarena.com, which is an online public resource for handset information.

However, starting in 2016, we see a new trend. The number of new chipsets available in the consumer market each year is diminishing. We analyzed the processor data, which we mined from gsmarena.com, and found two possible reasons. First, as Figure 3 shows, fewer companies are competing in the aggressive mobile consumer market. To gain a deeper level of understanding, we dissect the data in Figure 3 further and hone our analysis on a subset of the most widely known chipset manufacturers. As Figure 4 shows, Texas Instruments (TI) was producing a variety of OMAP processors between 2007 and 2012. But then, TI ceased the OMAP line. Similarly, Intel's foray into mobile computing was short-lived. After introducing the Atom processor in 2012, Intel ceased its mobile processor efforts in 2016. The same is true of other companies. Second, mobile vendors appear to be consolidating their effort into fewer, more-capable chipsets. For instance, in 2014, there were 49 Qualcomm chipsets, whereas in 2017, there were only 27. MediaTek also appears to be following a similar trend. We postulate that this is because most smartphones today require high-end capabilities. The exceptions are Apple and Samsung who have almost always focused on producing high-end consumer chipsets for premier devices.

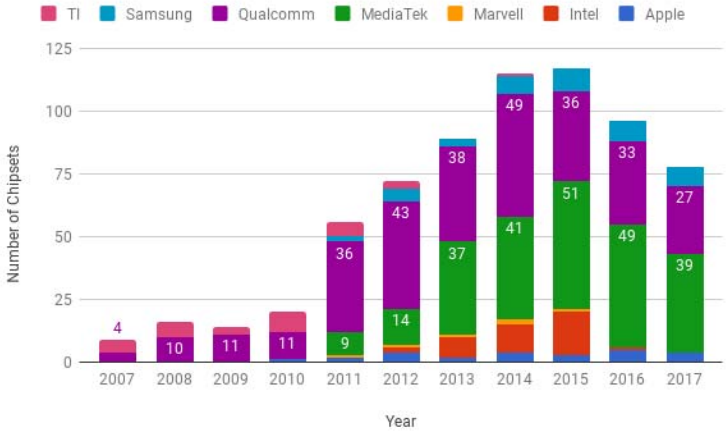

Figure 4. A breakdown of SoC chipsets from 2007 to 2017 for the most dominant industry players in the mobile handset consumer market.

Despite the decreasing trend in chipset diversity, we expect that processor customization will continue. The end of Dennard scaling coupled with Moore's law slowing down to a halt means

that processor customization is necessary to deliver expected generational improvements in hardware performance. In the foreseeable future, many believe that domain-specific processor customization is the only practically viable means to close the "specialization gap"[5] (the growing gap between hardware capabilities and applications' demand for more performance).

Customization has resulted in mobile processors evolving to have high core clock frequencies, aggressive microarchitectures, multicore designs, asymmetric architectures, heterogeneous execution, and domain-specific acceleration. Matthew et al. present a discussion about the evolution of mobile processors over the past decade.[6] Based on their findings, one can argue that mobile processors are on par with the complexity and orchestration needs of their desktop counterparts.

In practice, however, building a high-performance mobile device extends well beyond the capabilities of the processor. As Figure 5 shows, mobile computer architecture has a symbiotic relationship with users, applications, and the rest of the mobile form factor. The device form factor imposes strict power, energy, and thermal constraints. The processor architecture must operate within these confines to deliver good performance for applications. Users use these applications only if the processor provides a satisfactory user experience; no user wants a sluggish user experience even if the processor boasts outlandish capabilities and the application touts rich features. Therefore, users ultimately drive the features and capability requirements of future processors, which architects must strive to deliver in the face of increasing power consumption. This "virtuous" (sometimes "vicious") cycle of innovation is the source of many challenges in the industry.

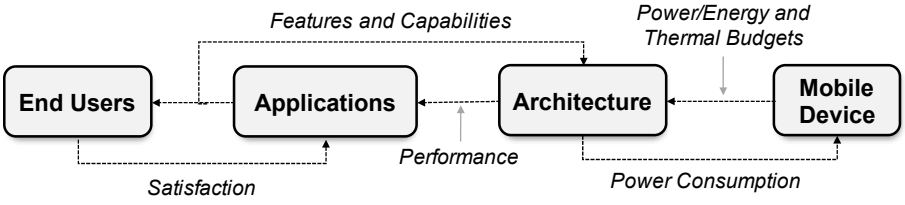

Figure 5. The "mobile device virtuous cycle" of development and innovation.

THE DEARTH OF MOBILE COMPUTER ARCHITECTURE RESEARCH

Sustaining the "mobile device virtuous cycle" warrants a holistic approach to designing mobile processors efficiently. We must understand the role of the processor architecture in the context of the whole system. However, the vast majority of mobile computer architecture work in industry and academia tends to focus on a subset of the picture illustrated in Figure 5. The focus is almost always on the siloed interaction between the applications and architecture. Architects optimize the system based on readily measurable observables, such as performance and energy, and fail to consider extraneous factors, such as user satisfaction and the device's imposing constraints.

To overcome the myopic view adopted by many, we require active research. But despite the overwhelming number of mobile processors "in the wild," in our pursuit to understand the state of mobile computer architecture research, we find that a disheartenedly small number of researchers pay attention to mobile computer architecture design. We searched through previously published literature and found that only 1 percent of all accepted papers (24 out of 1,502 papers) in HPCA, ISCA, MICRO, and ASPLOS combined (which are widely accepted as the "Tier 1" or "flagship" computer architecture conferences) focus on mobile computer architecture.

In fact, we uncover that the community pays more attention to processor architecture research for datacenter computing than for mobile computing. Figure 6 shows the number of accepted papers on mobile computing in each of the top conferences as compared to the number of papers on datacenters since 2007. While datacenter research has become mainstream in the conferences (the number of papers is on par with traditional topics such as cache optimization and core design), mobile architecture research is still very much in its infancy. The total number of papers in 2016 on mobile computing is smaller than that of datacenters in 2010.

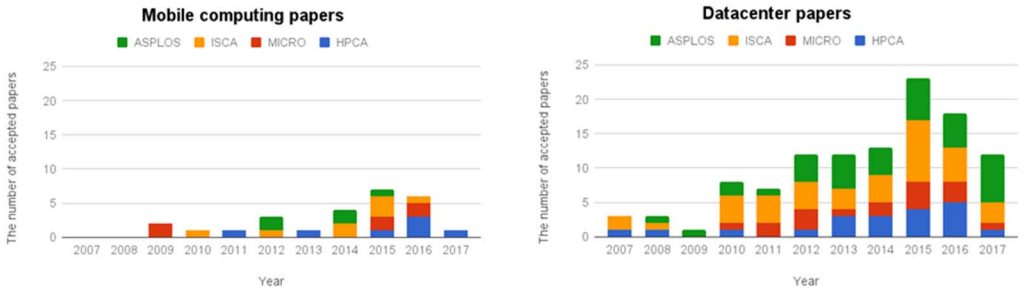

Figure 6. The number of accepted papers on mobile computing vs. datacenters from 2007 to 2017.

So why is it that only 1 percent of all architecture research papers published each year in the top computer architecture conferences focus on mobile computing? Surely it can't be that the architecture community thinks that mobile computing is less important than datacenter computing. There are more than two billion mobile devices. Taking all of the major cloud service providers—Google, Amazon, Microsoft, and Facebook—into account, a liberal estimate of the number of servers worldwide is perhaps 10 million. So, the current smartphone-to-server ratio is 200:1.

We believe that the dearth of research on mobile computer architecture fundamentally stems from the severe lack of knowledge about how mobile devices are used on a daily basis, how they perform "in the wild," and what challenges the industry is facing with mobile computer architecture design.

To this end, in this article, we discuss the reasons for the dearth of mobile computer architecture research from an industry perspective and focus on the challenges and opportunities as we look into the future. Understanding the mobile computer architecture challenges will enable us to comprehend not only the appropriate target workloads but also the metrics and methodologies to consider that can enable future findings. To fully understand and address all of these issues, we present our perspective in the form of "ten commandments" that ought to be observed in conducting mobile computer architecture research. We validate our commandment claims with data from the field, if possible, and, if not, we conduct experiments to justify the commandments.

The commandments strike at the heart of three primary categories: workloads (§1), metrics (§2), and methodology (§3). For each category, we present our opinions to help the community avoid common pitfalls. While a large number of our observations are based on smartphone computing principles, toward the end, we also discuss how they apply to other personal consumer devices.

The ten commandments of mobile computer architecture are:

1. Thou shalt not rely solely on benchmarks. (§1)
2. Thou shalt not cherry-pick applications. (§1)
3. Thou shalt not ignore the web browser. (§1)
4. Thou shalt not drop a frame. (§2)
5. Thou shalt not idolize microarchitectural efficiency. (§2)
6. Thou shalt not ignore tails in user experience. (§2)
7. Thou shalt not assume software is hardware-agnostic. (§3)
8. Thou shalt not disregard the IP blocks. (§3)
9. Thou shalt not turn a blind eye to energy and thermal. (§3)
10. Thou shalt not presume this list is complete. (§1,§2,§3)

WORKLOADS

1. Thou shalt not rely solely on benchmarks.

Numerous benchmarks have emerged to act as steadfast workloads for evaluating mobile devices. SPEC CPU, Geekbench, AnTuTu, and Quadrant are some of the most widely popular

benchmarks. Although SPEC CPU is generally targeted at desktop and server systems, efforts have been made to port the workload to evaluate it on mobile processors for evaluation purposes. Benchmarks like Geekbench and AnTuTu focus on core computational kernels (such as Dijkstra, JPEG compression, GEMM, and FFT) that exercise the CPU and GPU steadily.

In contrast to many of the steadfast benchmarks, which are composed of a handful of select applications, the mobile application ecosystem is diverse and the application's characteristics change overwhelmingly fast. After ten years of mobile computing, there are more than 3 million applications currently in the Google Play Store. The number of applications available first exceeded 1.5 million in 2015. This means more than 60,000 new applications are released in the Google Play Store each month. Keeping pace with the mobile ecosystem is challenging because benchmarks evolve more slowly than real-world applications. Take, for example, the continuously widening gap between SPEC CPU release dates: 1992, 1995, 2000, 2006, and 2017. Over the years, time between releases has lengthened despite workload advances in the real world.

The key workload challenge is the strong discrepancy between the performance characteristics of the real-world mobile applications and the mobile benchmarks. Many real-world applications are highly multithreaded and frequently communicate with other processes. Their performance characteristics are affected by the user's interactions. User satisfaction in the real world is determined in the order of milliseconds, on a per-event touch response latency rather than on computationally intensive steady state behavior. As a result, many of the steady state benchmarks fall short of faithfully capturing the interactive effects that are common on mobile devices.

Moreover, the benchmarks tend to undermine the importance of looking at the SoC as a whole and involve communication with other processes and several subcomponents, such as the ISP, IPU, and video/audio decoders and encoders, all of which contribute to the overall device performance and user experience. As we discuss later in the eighth commandment, understanding the role of the IPs and faithfully capturing holistic workload activity is important.

Mis-representativeness between real-world applications and benchmarks can have severe implications when making architectural design decisions. Architects might be misled to optimize for the wrong bottlenecks and make incorrect trade-offs if they misunderstand the consequences of using these benchmarks. Therefore, a strong call to action is the development of a benchmark suite that faithfully represents real-world application characteristics and user interactivity. But this in and of itself presents a new set of challenges, as the following two commandments state.

2. Thou shalt not cherry-pick applications.

Now that we have emphasized the focus on real-world applications, the next important step is to ensure that we pick the right applications to investigate deeply for architectural design decisions. It is typical practice in the community to select the top applications from the app stores to conduct detailed microarchitectural analysis studies. But architects must learn to be cognizant of the aforementioned pace of change, as the popularity of mobile applications can evolve quickly.

Figure 7 shows the popularity for three well-known applications: Google Chrome, Angry Birds, and Pokémon GO. The data presented here comes from Google Trends. Each of the three applications show distinctly different behavior over the course of time. We see a rapid decline in the interest in Angry Birds; it has dropped by about 80 percent over the past five years. Interest in Pokémon GO spiked briefly when the game introduced location-based and AR technology in mid-2016 but collapsed quickly. In contrast, Google Chrome shows less-dramatic shifts in popularity over time, holding a stable level of popularity over the course of nearly seven years.

However, Angry Birds is still a popular application for benchmarking mobile systems in the literature, and, as such, characterizing and optimizing the processor's architecture for this workload (that is trending out of popularity) will have little future impact. The conclusions drawn from studying old applications like Angry Birds might hurt processor architects when the design is shipped into products because they would be optimizing future processors for past workloads.

Therefore, it is important for hardware vendors and architects to keep pace with which applications are really "hot" and keep the mobile benchmarking application suite updated. Given that

SoCs take more than a year to build, it is important to project into the future and anticipate novel use cases. This is likely easier said than done, but we believe it can be achieved. For instance, if we take mobile gaming applications that rely on rendering engines as an example, we can anticipate the expected demands of future games by selecting the most challenging frames that the processor has to process in the rarest of all events and assume that they will be the common case.

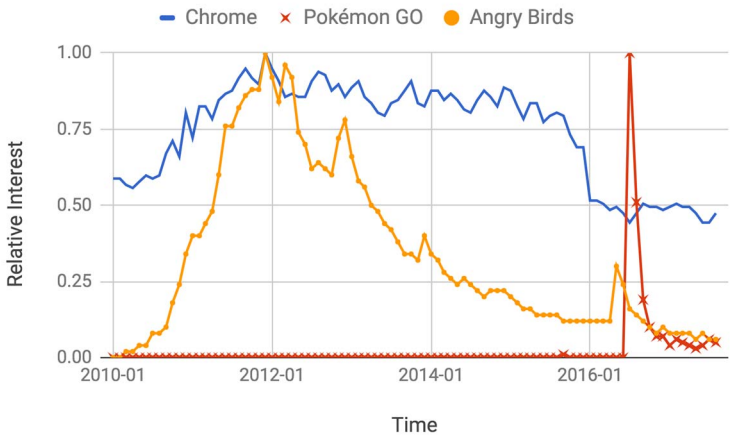

Figure 7. Changing application interest from 2010 to 2017.

3. Thou shalt not ignore the web browser.

As of August 2017, mobile was responsible for 52 percent of all Internet traffic,[7] and the gateway to the Internet is the web browser. Moreover, in many developing countries, users tend to prefer the browser over installing single-purpose or dedicated native applications, since web browsers are typically well optimized for dealing with variability in operating environments, such as intermittent network connectivity and data usage conditions. Therefore, the browser is a canonical application to include in any new benchmark suite and study on a mobile device.

In addition, many mobile applications, such as social and messenger applications, allow users to load webpages inside the application using so-called "WebView." A WebView allows developers to seemingly roll in their own web browser and display online content using the browser-rendering engine. This also allows developers to display webpages and include methods to navigate forward and backward through a history, zoom in and out, perform text searches, and more. This level of flexibility for deploying mobile applications further solidifies the browser's significance.

The browser's broad capabilities and flexibility are causing it to outpace many mobile applications in density and complexity. Figure 8 shows the change in Android Package Kit (APK) size for Google applications such as Gmail, Google Photos, Chrome, and YouTube. In a short period of two years, most applications have doubled. In contrast, Chrome has grown exceptionally fast. It has grown by eight times over the past five years. Chrome is six times larger than YouTube and Gmail and three times larger than Photos in its raw code and data footprint alone.

More than 70 percent of Chrome's APK is code and libraries, while the rest can be attributed to manifest files and other resources. The code contained within the APK exhibits highly irregular and complex control flows that stress virtually any mobile processor's capabilities. Since the browser, a single application, can render any of the billion webpages on the Internet, it can exhibit vastly different runtime code characteristics based on its input.[8] So, the performance of the browser-rendering engine plays an important role in the overall mobile device user experience.

Traditionally, it is believed that the mobile web browser is a network-bound (and network performance-limited) application. In other words, many assume that user experience while browsing is solely affected by network performance. However, advancements in LTE network and WiFi speeds have caused the browser to be more compute-bound.[9] As such, the browser relies heavily

on the CPU for its rendering and network processing. The browser typically spawns tens of threads that require a large amount of data and code footprints along with frequent inter-process (thread) communication.[10] Moreover, most browsers like Chrome rely on an asynchronous execution model that exercises nearly all of the IPs in a mobile SoC, including the CPU, GPU, video and audio decoders, networking, crypto-engine, and the various communication IP blocks.

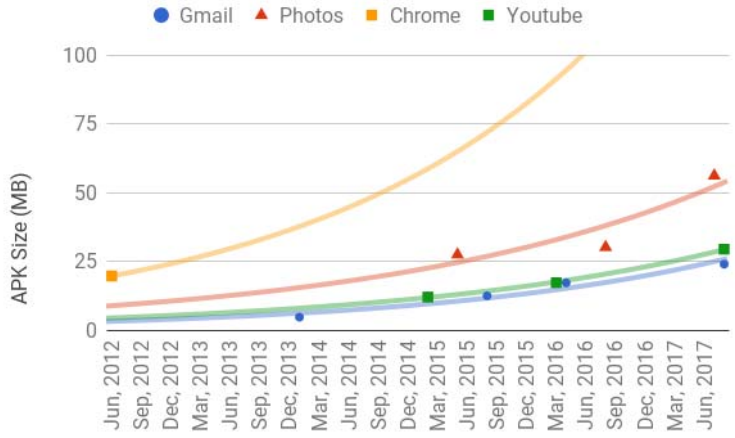

Figure 8. APK size increase from 2012 to 2017. Chrome was 157 MB in July 2017.

Mobile processor architects must include the browser in the creation of a benchmark suite and any sort of hardware or software optimization analysis. In many ways, the browser is akin to the GCC benchmark from the SPEC CPU benchmark suite, which has stood the test of time. GCC is one benchmark that has consistently retained its place through the evolution of the SPEC CPU suite, starting from CPU'92 and lasting all the way to the latest release of the CPU 2017 suite. The browser is similar to GCC in that it has also stood the test of time. Virtually every mobile device on the planet comes with a browser. So, it only makes sense to study the browser.

METRICS

4. Thou shalt not drop a frame.

Traditional architecture research focuses on hardware-centric metrics, such as instructions-per-cycle (IPC), cache misses-per-kilo-instructions (MPKI), and the runtime of an application. This works well for steady-state, streaming-style applications found in server class systems. However, except for video streaming (record or playback) applications, the majority of mobile applications are event-driven—the processor is waiting for user events. Events trigger task processing. So, it is important to evaluate mobile systems using event-driven characteristics that matter to the user.

An average user taps, types, swipes, or clicks her or his device 2,617 times a day, and about 10 percent of us perform those actions 5,427 times a day.[11] So, what matters to users is the touch responsiveness of the system (the time it takes to render a frame after a touch input). To ensure "buttery smooth" responsiveness, the system architecture as a whole must maintain 60 frames per second (FPS)[12] consistently without any dropped (or delayed) frames, commonly referred to as "jank." This proves challenging because maintaining 60 FPS means all processing (computing, networking, and rendering) must take place within 16.67 ms per frame.

As a general rule, it is better to design a system that can provide a sustainable throughput with no dropped frames than to build a system that has a high frame rate (such as 120 FPS) but drops frames even occasionally. For example, it is better to have an average frame time of 15 ms with no frames taking longer than 16 ms than to have 99 percent of the frames taking 14 ms and 1

percent taking 17 ms. Because in the latter scenario, a system would glitch once every two seconds! This sort of display jank is extremely noticeable to the sensitive human eye.[13]

Given the criticality of responsiveness for user experience, it would be prudent for mobile architectural simulators to report the frames that have been successfully processed and those that have been dropped per second, in addition to the more traditional raw runtime and microarchitectural efficiency metrics of the processor. Tools such as gfxinfo that are readily available and ship with the Android Open Source Project (AOSP) allow researchers to collect this type of user perceivable data on devices. Established simulators such as gem5 that are capable of full-system simulation already support this sort of capability. They simply need to be considered as a metric.

5. Thou shalt not idolize microarchitectural efficiency.

The textbook "Computer Architecture: A Quantitative Approach" by David A. Patterson and John L. Hennessy has taught every architecture student to have a strong and quantitatively rigorous approach to measuring simulated or real hardware performance. But a lot has changed over the past 10 years since the arrival of the smartphone. The measure of performance in a mobile device is not how fast a processor can compute; rather, its true capability lies in its ability to deliver user-perceivable satisfaction improvements. For instance, doubling the TLB from 32 to 64 entries or increasing CPU clock frequency might seem like the right trade-off to improve performance at the expense of power consumption. In practice, however, if the microarchitectural enhancements cannot translate to measurable user experience, doing so won't be useful.

As an example, Figure 9 shows the cumulative distribution function (CDF) for frame rendering latency for the "Invalidate" test in UiBench. We consider 8,000 sample frames. We vary the CPU clock frequency of a Google Pixel XL device, which consists of asymmetric cores, to evaluate the impact on consistently maintaining a fixed FPS rate. Configurations with lower frequency gradually show longer rendering latency per frame on average and much longer, noticeable tails. For example, the fastest configuration (blue line) shows a 5-ms gap between the maximum and minimum rendering latency, whereas we notice a more-than-15-ms gap for the slowest configuration (green line). But if we assume a 60 FPS VSync rate, the data indicates that there is an opportunity to slack in the computation and save power. By dropping from 2.15 GHz to 1.67 GHz, we can still meet the 16.67-ms cut-off for all the frames, save a significant amount of power and energy, and run cooler. Put another way, increasing the frequency from 1.06 GHz to 1.67 GHz is worth the extra energy, as it helps meet the cut-off latency. Pushing beyond 1.67 GHz can be wasteful in terms of performance and energy. There is an interesting trade-off here: In terms of overall power efficiency *and* user experience, are high-performance, high-power cores better than slower, low-power cores that just meet the latency requirement?

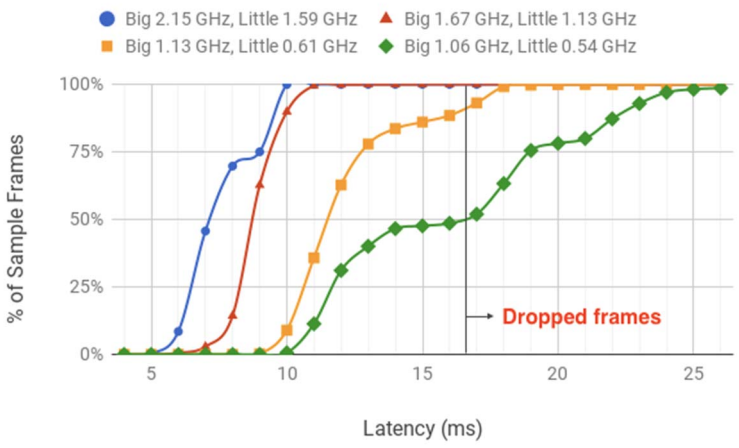

Figure 9. Distribution of frame rendering latency for the UiBench application for different CPU clock frequencies.

Therefore, it is important to understand the relationship between microarchitectural enhancements and the actual user impact. Zhu et al. provide a model to think about performance in the form of "perceptibility," "tolerability," and "un-usability."[14] It is possible that a systematic construct can be useful in rationalizing about the benefits of certain optimizations. But without such approaches, it is hard for architectural design-time decisions to have real-world product impact.

6. Thou shalt not ignore tails in user experience.

Tail latency is a well-known problem in data centers.[15] It occurs when the overall response time for a request is dominated by a long tail distribution because of multiple servers operating in parallel to service the request. But little do people realize that mobile applications also suffer from long tail latency issues. There are multiple sources of variation in a mobile device: thermal throttling, code interpretation, dynamic recompilation, garbage collection, background killing of applications due to memory pressure, nondeterministic networking and communication, Android device differences, kernel scheduling differences, and so on. The list is long. Ultimately, all of these sources of variation affect user experience because users recall "long tail" experiences.

Figure 10 shows a histogram of response times for the onCreate user event from two first-party Google applications, both of which have at least a few hundred million users. The curves for the two applications are different because the activity corresponding to the event is different between the two applications. However, both applications show a long tail. The average response time is around 500 ms, though the tail can be more than eight times as long (> 4,000 ms).

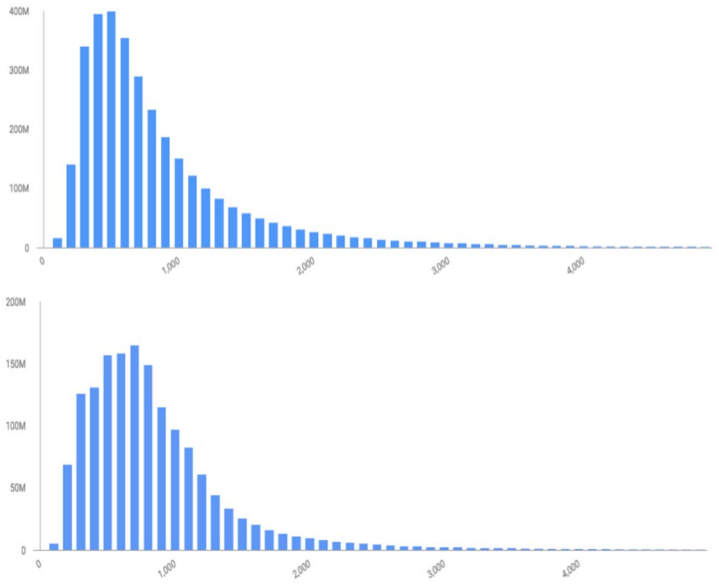

Figure 10. Distribution of latencies for an event in two widely used first-party Google applications based on field data.

The precise reasons for the variance we observe in Figure 10 are hard to pinpoint given the nature of the complex Android OS. Controlling for such variability is challenging, yet important. At present, there is no well-known way to control this variability, because there are too many sources. This means that to have confidence in whether any architectural optimization has impact, the optimization must first withstand the variance in the execution times. Otherwise, we do not know if the improvement is truly from the experiment or if it's by chance. So, often, the safest way is to report the distribution or assume the worst case and report results accordingly.

METHODOLOGY

7. Thou shalt not assume software is hardware-agnostic.

Mobile applications are written with user experience and hardware capabilities in mind. There are more than 24,093 distinct Android devices in the world, and the performance capabilities of these devices vary drastically.[16] The Android OS has built-in capabilities for applications to take different execution paths (silently) based on a queried hardware feature. Such functionality allows some hardware to be restricted from sophisticated features by popular applications because parts of the application might not be fast enough to provide responsive user experience.

We use Chrome as an example to illustrate the extent of such "software specialization." Consider the real-world case of the Chrome browser dealing with video encoding. Different mobile devices support different codecs (or codec variations) in their hardware decode engine, and so the browser must adapt to them, with a fallback to software decoding in the event that the codec used by a video isn't supported in hardware. Another example involves different GPUs. Some GPUs expose different extensions over the base OpenGL ES, and applications like Chrome might use different rendering algorithms based on the presence, or absence, of particular hardware extensions. For example, NVIDIA GPUs support a proprietary "path rendering" extension that the Chrome browser will use if the GPU supports it, which reduces 2D rendering costs for curves. But if it is not present, the browser takes a completely different software code path.

So, one cannot assume that a given workload runs identically on different devices. Given the extent of heterogeneity in the mobile hardware ecosystem, one needs to understand that the same application can be hard-coded differently on different devices to perform differently. Because hardware-aware software tuning isn't rare in the mobile ecosystem, when we perform comparative performance analysis between any two platforms, we need to be aware of such crucial differences. Sadly, however, this requires an inscrutable level of domain-specific application level knowledge. But that is the price to pay for working in a rich and highly heterogeneous system.

One cannot assume that a given workload runs identically on different devices. Given the extent of heterogeneity in the mobile hardware ecosystem, one needs to understand that the same application can be hard-coded differently on different devices to perform differently.

8. Thou shalt not disregard the IP blocks.

The mainstream architecture community has long focused on general-purpose CPUs and GPUs. However, unlike CPUs and GPUs that have been optimized for decades, SoCs are, by definition, a modular collection of special-purpose compute, communication, and storage IP units. In a high-performance SoC, the majority of the die area is dominated by IP blocks. The Apple A8 SoC supposedly has close to 30 IP blocks, and the application processor occupies less than 20 percent of the total die area.[17] Qualcomm also touts similar claims: "[O]n a modern Snapdragon processor, less than one-third of the total silicon real estate is taken up by the CPU."[18] Many of the IP blocks are typically licensed from third-party companies and assembled together through standard interfaces (such as ARM AXI/ACE bus) to meet a targeted purpose.

In an SoC, multiple IP blocks are active at the same time and communicate frequently with each other over the Network-on-Chip (NOC) fabric. These interactions prove particularly challenging for architects to design for because they often manifest in less-than-expected performance post-tapeout when the chip is running realistic use cases and the IPs are operating concurrently. Often, however, the IPs are studied in isolation. In reality, the IPs are concurrently competing for various resources when they are deployed to handle an everyday, realistic use case scenario.

To understand the significance of studying multiple IP flows concurrently, consider the typical use case of recording a 4K display resolution video. Figure 11 shows the dataflow between the various IP blocks. Compressed raw frame data from a camera-sensor feeds into the Image Signal Processor (ISP) that performs pixel processing and writes the frame out to DRAM. The frame data then diverges into two streams: a "preview stream" to the display and a "video stream" to the storage subsystem. The preview data stream is sent from the ISP to the display pipeline through the DRAM. The display pipeline handles the UI layers compositing (both the display controller and the GPU do compositing depending on layer count), and they finally render the preview to the screen. The video stream in the ISP is compressed and undergoes several additional passes through the GPU. The compressed ready frames are delivered to the video encoder buffer in DRAM. The video encoder processes multiple frames at a time and writes the encoded compressed stream to DRAM where the CPU ultimately streams it to storage. A real-time audio stream is routed through the DSP and encoded into the video stream for persistent storage.

Hopefully, this everyday use case illustrates the complexity and heterogeneous processing in a mobile system. Furthermore, the magnitude of concurrency in a mobile system should reinforce the need for computer architects to understand the system as a whole, and not focus solely on the CPU subsystem. The brunt of the work in a mobile device is handled by the IP blocks, and the CPU generally plays a minimal role, mostly coordinating activity between the different IPs.

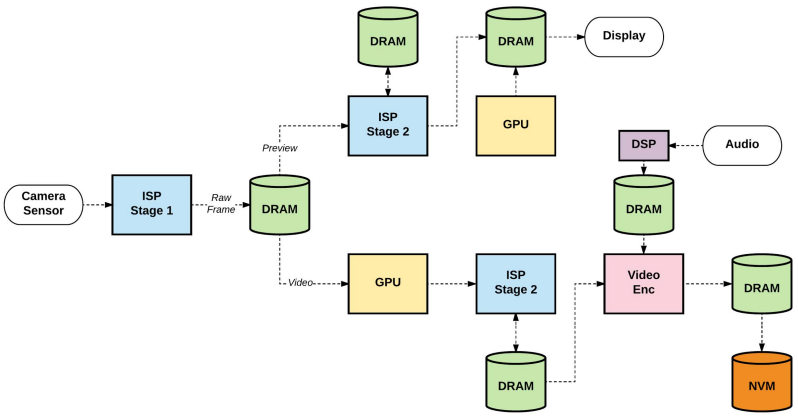

Figure 11. This example of a 4K, 60 FPS video capture use case shows the importance of considering IP flows. 4K refers to a horizontal screen display resolution in the order of 4,000 pixels.

Therefore, architects are strongly encouraged to consider the role of the various IP blocks in the process of evaluating a mobile system. There are numerous opportunities for research and development of novel solutions to enhance mobile processor performance. For instance, consider the number of memory streams that are concurrently active in the mobile chipset in the 4K video example above. One could envision new caching policies to minimize the overall DRAM traffic, develop point-to-point communication channels between the various IPs, and so forth. Solutions can be developed to either enhance performance of the system or lower its power consumption.

9. Thou shalt not turn a blind eye to energy and thermal.

No discussion on mobile is ever complete without stating the obvious. Mobile devices operate under strict thermal and energy budgets. But perhaps a little less-known fact is that there is no Moore's law for batteries.[19] Battery density doesn't double until every ten years, while Moore's law continues every two years. As a result, we witness increasing battery sizes in smartphones, which directly translates to capacity. The Samsung Galaxy S8 (2018) has a 3,000 milliampere hour (mAh), while the Samsung Galaxy S (2009) had a 1,500 mAh. The five-fold discrepancy between battery density improvements and Moore's law implies there is a steadily widening gap between compute requirements and energy availability in a mobile form factor.

It is easy to tout performance improvements from different microarchitectural solutions without fully considering the consequences. However, we should consider ourselves warned because passive cooling and limited battery power impose severe restrictions on the device's capabilities. Cooling is a major problem because all humans have a low tolerance for high surface temperatures. We can handle at most 45 degrees Celsius before our skin starts experiencing pain.[20] With the end of Dennard scaling, rising power density is a serious issue. Increasing power density is leading to thermal challenges that far exceed a device's inherent passive cooling capability.

To overcome this issue, it is important to understand that energy consumption and thermals in a mobile device are highly subjective to real-world use cases, and, therefore, diverse use cases must be considered altogether to comprehensively evaluate the power efficiency of a mobile device. For example, consider the power consumption of the SoC under two common, but extremely different use cases: web browsing and 4K video recording. Figure 12 shows the SoC power distributions of these use cases, which indicates that Chrome is a CPU-intensive application, while the video recording heavily employs many other IPs on the SoC (as shown previously in Figure 11). The SoC consumes five times more power for 4K video recording (3,473.55 mW) than for running Chrome and scrolling through a locally cached website (651.77 mW).

The differences in the power consumption and activation of different IP blocks indicates that we need to optimize the system as a whole by considering power, thermal, and performance together. We expect this trend will continue as more IP blocks and custom accelerators are integrated into the SoC for emerging use cases, such as machine learning and augmented reality. Therefore, measuring and quantifying the power peaks and energy consumption of new ideas is necessary given that battery and heat dissipation are first-order constraints for mobile systems.

It is important to understand that energy consumption and thermals in a mobile device are highly subjective to real-world use cases, and, therefore, diverse use cases must be considered altogether to comprehensively evaluate the power efficiency of a mobile device.

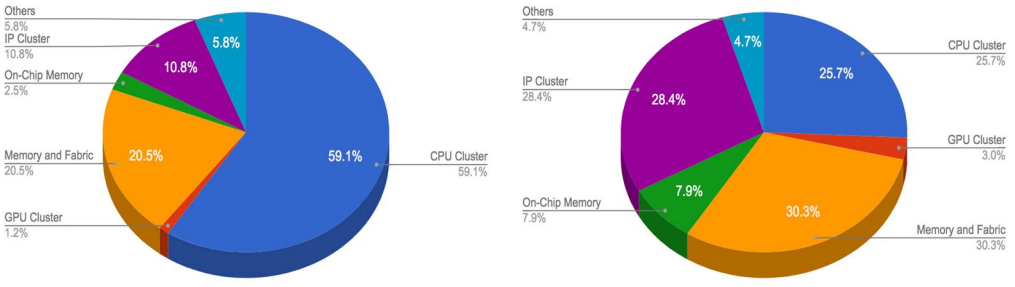

Figure 12. Power distribution differences for two common mobile device use cases. (left) Web browser scrolling with a total power consumption of 651.77 mW. (right) 4K video recording with a total power consumption of 3,473.55 mW.

10. Thou shalt not presume this list is complete.

Mobile computing is a rapidly and continuously evolving ecosystem, and the hardware requirements will need to evolve as time progresses. It would be presumptuous for anyone to assume that any one set of commandments can ever be complete. To this end, the tenth commandment

acknowledges that the current list is by nature incomplete and that the aforementioned commandments should be improved upon over time as mobile computing evolves.

A CALL TO ACTION

Mobile computing is here to stay, and it presents unique challenges that are different from server and desktop systems. The goal of this article is to motivate the community to address these challenges. Here, we present some recommendations for mobile architecture research based on the commandments that span the three important areas: workloads, metrics, and methodology.

Workloads (commandments 1 through 3): We make a case for using real workloads to drive the performance evaluation of mobile devices. But real workloads are complex and need to be dissected further to gain deeper insights. We recommend splitting a workload into smaller parts called regions of interest (ROIs). An ROI is a slice or portion of a workload that represents an interesting aspect that needs to be studied carefully. For example, application startup is a specific ROI because it indicates a device's capability and impacts user experience. An application may have multiple ROIs. For example, a user opens Chrome and wants to search for an article. There are many stages. Each of the different actions (such as Chrome startup, typing a keyword, scrolling, and hitting one of the search results) can be qualified as a distinct ROI. These distinct ROIs need to be isolated and studied in greater detail under improved architectural simulators.

Metrics (commandments 4 through 6): We state that user experience should be measured using jank and long tails in addition to traditional microarchitectural metrics. But there are several other metrics that also matter, including application startup latency (the time from clicking the icon to the first interaction with the application), tab switching latency (when a user wants to switch from one foreground task to another), and touch latency (the time it takes for the user to notice a difference on the screen as a result of his or her touch-screen input). Some metrics, such as startup latency, can be directly measured using existing benchmarks and applications. However, other metrics, such as tab switching latency, can't be captured using existing workloads. We must construct benchmarks to isolate and capture such effects. Understanding these different "phases" of application behavior can have a direct consequence on the architectural design.[21]

[Some] metrics, such as tab switching latency, can't be captured using existing workloads. We must construct benchmarks to isolate and capture such effects.

Methodology (commandments 7 through 9): We emphasize the need for systematic evaluation. Benchmarking mobile devices is tricky due to the interactivity. Mobile application ROIs are small (for example, responsiveness to a button click), so measurement error can be large if care isn't taken to benchmark systematically. Therefore, a key requirement for benchmarking is repeatability, which restricts workload variability. The goal for each "curated" workload ought to be to craft the workload in such a way that all or some parts of the workload can be run with strong repeatability on a variety of platforms, including simulators and real devices.

Simulation platforms are also an important requirement. Mobile simulators need to be of a different breed than the ones that have driven architecture research over the past two decades. SoC simulators need to be able to boot up-to-date Android, support a rich set of IP blocks, and interface with sensors. Detailed cycle-level models are likely to be less useful for initial fast and rapid design space exploration studies. Instead, we need a new breed of mobile architecture simulators, which need to be integrated with abstract models of the various IP blocks and network fabrics to help analyze complex use cases and their data flows (such as 4K video recording).

CONCLUSION

Mobile computer architecture is still in a nascent stage despite the proliferation of handheld devices. It is a rapidly evolving ecosystem, making it challenging to establish benchmarks, metrics, and experimental methodologies. But given its prevalence, it's paramount that these issues be addressed by both academia and industry. The ten commandments we present help identify many of the issues facing mobile architecture research, although it would be presumptuous to assume that any one set of commandments can be comprehensive and complete enough to capture the space in such a fast-moving environment. Therefore, the commandments should be improved upon over time, and the existing commandments taken with a grain of salt. The call to action for the community is to develop (and release) the vessels needed to improve today's ad-hoc practices. We should be able to evaluate realistic workloads using meaningful metrics and develop new methodologies that allow us to optimize complex use cases across the entire SoC, not just the CPU/GPU. We hope that the issues discussed here serve as a seedling for deeper research.

Many of the things discussed here are based on lessons learned from touch screen-based interfaces over the past decade. However, consumer devices are on the cusp of a new evolution involving head-mounted displays that have more stringent response time latency requirements than smartphones. Meanwhile, the medium for interaction is also changing. In the coming decade, speech will likely become an important modality for interacting with consumer devices. The community needs to think about how it can improve simulation and research infrastructure to new heights so that it can investigate these complex systems across all layers of the system stack.

REFERENCES

1. P. Sundararajan, *Google I/O '17 Keynote*, 2017; youtube.com/watch?v=Y2VF8tmLFHw.
2. *Ericsson Mobility Report*, report, 2017; ericsson.com/assets/local/mobility-report/documents/2017/ericsson-mobility-report-june-2017.pdf.
3. *IDC Worldwide Quarterly Server Tracker*, report, 2017; idc.com/tracker/showproductinfo.jsp?prod_id=7.
4. *ARM Holdings 2015 Annual Report*, report, 2015; annualreports.com/Company/arm-holdings-plc.
5. L. Ceze, M.D. Hill, and T.F. Wenisch, *Arch2030: A Vision of Computer Architecture Research over the Next 15 Years*, white paper, Computing Community Consortium, 2016; arxiv.org/abs/1612.03182.
6. M. Halpern, Y. Zhu, and V.J. Reddi, "Mobile CPU's rise to power: Quantifying the impact of generational mobile CPU design trends on performance, energy, and user satisfaction," *2016 IEEE International Symposium on High Performance Computer Architecture* (HPCA 2016), 2016, pp. 64–76; computer.org/csdl/proceedings/hpca/2016/9211/00/07446054-abs.html.
7. *Percentage of mobile device website traffic worldwide from 1st quarter 2015 to 3rd quarter 2017*, report, Statista, 2017; statista.com/statistics/277125/share-of-website-traffic-coming-from-mobile-devices/.
8. L.C. Greene and J.D. Hardy, "Adaptation of thermal pain in the skin," *Journal of Applied Physiology*, vol. 17, no. 4, 1962, pp. 693–696; physiology.org/doi/abs/10.1152/jappl.1962.17.4.693.
9. Y. Zhu, M. Halpern, and V.J. Reddi, "The Role of the CPU in Energy-Efficient Mobile Web Browsing," *IEEE Micro*, vol. 35, no. 1, 2015, pp. 26–33; computer.org/csdl/mags/mi/2015/01/mmi2015010026-abs.html.
10. C. Cascaval et al., "Concurrency in Mobile Browser Engines," *IEEE Pervasive Computing*, vol. 14, no. 3, 2015, pp. 14–19; computer.org/csdl/mags/pc/2015/03/mpc2015030014-abs.html.
11. M. Winnick, "Putting a Finger on Our Phone Obsession," *dscout blog*, blog, dscout, 2016; blog.dscout.com/mobile-touches.
12. B.F. Janzen and R.J. Teather, "Is 60 FPS better than 30?: the impact of frame rate and latency on moving target selection," *CHI '14 Extended Abstracts on Human Factors in Computing Systems* (CHI EA '14), 2014, pp. 1477–1482; dl.acm.org/citation.cfm?id=2581214.

13. I.S. MacKenzie and C. Ware, “Lag as a determinant of human performance in interactive systems,” *Proceedings of the INTERACT '93 and CHI '93 Conference on Human Factors in Computing Systems* (INTERCHI'93), 1993, pp. 488–493; dl.acm.org/citation.cfm?id=169431&CFID=850493721&CFTOKEN=65193828.
14. Y. Zhu, M. Halpern, and V.J. Reddi, “Event-based scheduling for energy-efficient QoS (eQoS) in mobile Web applications,” *2015 IEEE 21st International Symposium on High Performance Computer Architecture* (HPCA 2015), 2015, pp. 137–149; computer.org/csdl/proceedings/hpca/2015/8930/00/07056028-abs.html.
15. J. Dean and L.A. Barroso, “The tail at scale,” *Communications of the ACM*, vol. 56, no. 2, 2013, pp. 74–80; dl.acm.org/citation.cfm?id=2408794.
16. *Android Fragmentation Visualized*, report, OpenSignal, 2015; opensignal.com/legacy-assets/pdf/reports/2015_08_fragmentation_report.pdf.
17. Y.S. Shao et al., “The Aladdin Approach to Accelerator Design and Modeling,” *IEEE Micro*, vol. 35, no. 3, 2015, pp. 58–70; computer.org/csdl/mags/mi/2015/03/mmi2015030058-abs.html.
18. *ARM AND QUALCOMM: Enabling the Next Mobile Computing Revolution with Highly Integrated ARMv8-A based SoCs*, white paper, ARM and QUALCOMM, 2014; arm.com/files/pdf/ARM_Qualcomm_White_paper_Final.pdf.
19. F. Schlachter, “No Moore’s Law for batteries,” *Proceedings of the National Academy of Sciences of the United States of America* (PNAS), vol. 110, no. 14, 2013; pnas.org/content/110/14/5273.full.
20. Y. Zhu and V.J. Reddi, “High-performance and energy-efficient mobile web browsing on big/little systems,” *2013 IEEE 19th International Symposium on High Performance Computer Architecture* (HPCA 2013), 2013, pp. 13–24; computer.org/csdl/proceedings/hpca/2013/5585/00/06522303-abs.html.
21. H. Kim et al., “Accelerating Application Start-up with Nonvolatile Memory in Android Systems,” *IEEE Micro*, vol. 35, no. 1, 2015, pp. 15–25; computer.org/csdl/mags/mi/2015/01/mmi2015010015-abs.html.

ABOUT THE AUTHORS

Vijay Janapa Reddi is a visiting research scientist at Google working in the Mobile SoC Architecture team. He is on leave from his associate professor position at the University of Texas at Austin. His research interests include system performance, user experience, energy efficiency, and reliability for consumer devices and autonomous systems. He has a PhD in computer science from Harvard University. Contact him at vj@ece.utexas.edu.

Hongil Yoon is an SoC performance architect at Google. His research interests include computer architecture and operating systems in virtual memory and memory systems. He has a PhD in computer science from the University of Wisconsin-Madison. Contact him at hongilyoon@google.com.

Allan Knies is the simulation and performance lead in Google’s hardware division. Previously, he worked at Intel, where he contributed to the first-generation IA-64 processor, the Intel Research Berkeley Lab, and the US Department of Energy FastForward program. He has a PhD from Purdue University. Contact him at aknies@google.com.