UNIVERSITY OF CALIFORNIA, SAN DIEGO

Characterizing and Leveraging Processor Variability in Mobile Devices for Energy Efficiency

A thesis submitted in partial satisfaction of the requirements for the degree Master of Science

in

Computer Science

by

Roshni Chandrashekhar

Committee in charge:

Yuvraj Agarwal, Chair
Rajesh Gupta
Puneet Gupta
Geoffrey Voelker

2013

The Thesis of Roshni Chandrashekhar is approved and is acceptable in quality and form for publication on microfilm and electronically:

Chair

University of California, San Diego

2013

DEDICATION

To Amma, Appa and Chetan, it's time for another adventure.

EPIGRAPH

Nothing begins, and nothings ends,
That is not paid with moan,
For we are born in other's pain,
And perish in our own.
– Francis Thompson, Daisy, 1893

TABLE OF CONTENTS

Signature Page iii

Dedication iv

Epigraph v

Table of Contents vi

List of Figures viii

List of Tables x

Acknowledgements xi

Abstract of the Thesis xii

Chapter 1 Introduction 1
1.1 Motivation 2
1.2 Thesis Organization 4

Chapter 2 Related Work 6
2.1 A fluid hardware-software interface 6
2.2 Sources of Variability 7
2.3 Characterization of Variability 8
2.4 Adapting for Variability 9
2.5 Energy Savings on Mobile Devices 12
2.5.1 Instrumentation 12
2.5.2 Adapting for Energy Efficiency 13

Chapter 3 Experimental Infrastructure 15
3.1 Measurement System 15
3.1.1 Hardware 16
3.1.2 Software 18
3.2 Measurement Parameters 19
3.3 Experiment Workflow 23
3.3.1 Eliminating Measurement Errors 25

Chapter 4 Processor Power Variability in Mobile Devices 27
4.1 CPU Subsystem Variability 27
4.1.1 Applications 27
4.1.2 Observed Variability 28
4.2 Impact of Ambient Temperature 32

4.3 Discussion 35

Chapter 5 The Case for Software Adaptability 40
5.1 Application 40
5.2 Potential for Adaptation 44
5.2.1 Sample Scenarios 46

Chapter 6 Implications, Future Work and Conclusion 52
6.1 Implications 52
6.1.1 Instrumentation 52
6.1.2 Variability-Aware Adaptation 53
6.1.3 Service Provider Adoption 53
6.1.4 Potential User Interactions 54
6.2 Future Work 54
6.2.1 More Adaptive Applications 54
6.2.2 Analyzing Graceful Degradation 55
6.3 Conclusion 55

Appendix A Additional Figures and Observations 57
A.1 Power Measurements For 8 Channels 57
A.2 Run to Run Variation 58
A.3 Encodings and Processor Frequencies 59
A.4 Calculation of Margin of Error 60

Appendix B Other Workflow Details 61
B.1 NSF Demo Documentation 61
B.1.1 Software Packages 61
B.1.2 Workflow 61
B.2 ADB Commands 62
B.3 Video Quality Measurements 63
B.3.1 Using ffmpeg 63
B.3.2 BVQM 64

Bibliography 65

LIST OF FIGURES

Figure 1.1. Circuit variability as predicted by ITRS 3

Figure 2.1. Sleep Power Variability in ARM Cortex M3 Microprocessors 9

Figure 2.2. Power variability in DRAMS with identical hardware specifications manufactured by different vendors 10

Figure 2.3. Power variability in Intel i5-540M Processors at 2.53GHz. 11

Figure 3.1. Experiment Infrastructure Overview 17

Figure 3.2. WQEPM Web UI Screenshot 20

Figure 4.1. Power Usage Across Devices for LINPACK 31

Figure 4.2. Power Usage Across Devices for Whetstone 31

Figure 4.3. Power Usage Across Devices for Dhrystone 32

Figure 4.4. Total Power across devics for CPU Intensive Applications 33

Figure 4.5. Effect of Temperature on Device Variability - CPU-intensive Benchmarks 35

Figure 4.6. Effect of Temperature on Device Variability - Video Playback . . . 37

Figure 4.7. Idle Power Consumption Across Devices 38

Figure 5.1. Power versus PSNR for a high resolution video 43

Figure 5.2. Power versus PSNR for a medium resolution video 44

Figure 5.3. Comparing Power versus PSNR for two frame resolutions: Scenario 1 47

Figure 5.4. Comparing Power versus PSNR for two frame resolutions: Scenario 2 48

Figure 5.5. Power versus Frame Resolution for the *Tea House* video 50

Figure A.1. Power Measurements for 8 Channels for the Lowest Power Device 57

Figure A.2. Power Measurements for 8 Channels for the Highest Power Device 58

Figure A.3. Run to run variation 59

Figure A.4. Encodings and Phone Processor Frequencies 60

LIST OF TABLES

Table 3.1. Snapdragon MSM8660 Mobile Development Platform (MDP) 16

Table 3.2. Power Measurement Points 22

Table 4.1. Device Variability at 192MHz 29

Table 4.2. Per-Channel Variability at 192MHz 30

Table 4.3. Device Variability at 1.5GHz 30

Table 4.4. Per-Channel Variability at 1.5GHz 30

Table 4.5. Device variability at 1.5GHz at different temperature ranges for CPU-intensive benchmarks 36

Table 4.6. Device variability at 1.5GHz at different temperature ranges for video playback 36

Table 5.1. Video Details 42

Table 5.2. Processor power variability for video playback at different frame resolutions at 1.5GHz 45

Table 5.3. Difference between playing CIF and QCIF frame resolutions for different encodings 49

ACKNOWLEDGEMENTS

I thank Doctor Yuvraj Agarwal for his support as my advisor, guide, and the chair of my committee. His encouragement to focus on building the concrete from the abstract and his patience through the arduous terrains of proprietary information serve as the pillars for this work.

I also acknowledge Professor Rajesh Gupta for introducing me to the Variability Expeditions, and Professor Puneet Gupta who has been one of the biggest supporters of my project, and was always available to run things by despite his other commitments and remote location.

I thank Kiran Rudramani, Fred Bontemps and the other members of the Qualcomm Innovation Center Inc. for all the help with the infrastructure and devices, as well as the answers to my persistent questions. For help with Video Quality Measurements, I thank Aashish Pant, one of Puneet's former students and Margaret Pinson, a Co-Chair of the HDTV project in the Video Quality Experts Group (VQEG).

I would also like to thank Professor Geoff Voelker, who's been a wonderful mentor through my journey at grad school, and was always available for advice and help on all fronts.

For always being on my side, putting up with my insanity, and for sticking up for me through thick and thin, I thank my fabulous five, Manoj, Amal, Mukanth, Venmathi and Rakesh.

ABSTRACT OF THE THESIS

Characterizing and Leveraging Processor Variability in Mobile Devices for Energy Efficiency

by

Roshni Chandrashekhar

Master of Science in Computer Science

University of California, San Diego, 2013

Yuvraj Agarwal, Chair

As semiconductor manufacturers build smaller components, circuits and chips at that scale become less reliable and more expensive to produce, and no longer conform to the rigid hardware specifications usually expected of them. While traditionally, the onus of handling variability has been on the hardware manufacturers, there has been a recent push towards embracing device variability, especially in software, rather than hiding it by increasing guardbands applied to chip designs. Our work takes a higher-level software systems approach to previous studies of embedded sensing systems made in this regard and extends them to mobile devices, mainly smartphones, which are the current

generation of general purpose computing devices.

We begin by measuring and characterizing processor power variability in mobile devices through fine-grained power measurements on a suitably instrumented platform. We observe variation in processor power consumption ranging from 6% to 15% across smartphones that are manufactured to be identical. This variability, if harnessed properly, could convert into improvements in battery lifetime of 30 to 70 minutes. In this thesis, we also make the case for adaptive software that can leverage information about patterns of variability observed across devices for improved energy efficiency. Using video playback with different tunable parameters as a motivating example, we discuss the trade-off between quality of service and energy usage an the role device variability can play in these trade-offs and find that it is possible for us to proactively choose the encoding and frame resolution parameters to use for video playback, resulting in estimated energy savings of 3-15% which translates to improved battery lifetime of an hour.

Chapter 1

Introduction

Computer systems have been reaping the benefits of Moore's Law driven scaling of the semiconductor manufacturing processes for the last three decades. However, we are now beginning to see dramatically deteriorating effects of material properties on the active and leakage power, which in turn, results in increased variability across components. For now, dealing with this problem was limited to hardware manufacturers who respond with increasing guardbands applied to microelectronic chip designs, thereby continuing to conform to the outward appearance of a rigid hardware specification. However, as this problem continues to scale with shrinking critical dimensions of chip designs, the cost of maintaining the facade of a rigid specification through precise control over manufacturing quality has translated into exponentially increasing costs of fabrication and equipment.

This variability in microelectronic manufacturing manifests itself in many forms, such as variation in the threshold voltages of transistors in a chip, which in turn affects the power consumption and the maximum operating frequency of the resulting chips. Besides the primary source of semiconductor manufacturing, variability may also manifest through some other secondary factors or sources like the environment or operating conditions, total energy capacity of batteries, transistor aging and multi-sourcing of parts with identical hardware specification from different vendors [6, 45]. With this knowledge of variability in process manufacturing, we intend to explore, analyze and characterize

the processor power variability in mobile devices, which are rapidly becoming the most prominent general purpose computing platforms. The next few sections will present the underlying motivation for this thesis and the issues addressed through this work.

1.1 Motivation

The NSF Variability Expedition [1] makes the case for observing hardware variability across devices, due to wear-out over time or changing environmental parameters like temperature and exposing such observed behavior to the software stack so as to save the expense of manufacturing hardware that meets rigid specifications. It has been observed that this conformity is more expensive than allowing diversity to flourish, and the Expedition envisions a future where embracing the diversity of devices and hardware components to within the boundaries of the software stack makes for more robust and power-saving computing devices.

Gupta et. al. [16] present the basic motivation for a flexible hardware-software stack that can take advantage of a relaxed hardware design. The authors also advocate the case for underdesigned hardware in the light of increasing variability in manufacturing and the expenses borne out of such variability in the form of the proposed Underdesigned and Opportunistic (UnO) computing machines. The ITRS predictions [22] of circuit variability in Figure 1.1 show that the trend is set to continue even further, thus laying the ground for underdesigned hardware that can be used in combination with a flexible software stack to dynamically adapt to variability. While the primary motivating factor for exploring and analyzing variability has been limited to hardware manufacturing, in this thesis, we approach variability through a higher level systems perspective by the detailed characterization of variability in power consumption across multiple mobile devices and mechanisms to harness this variation for improving device battery life.

With over 480 million devices sold in 2011, smartphones are now more than

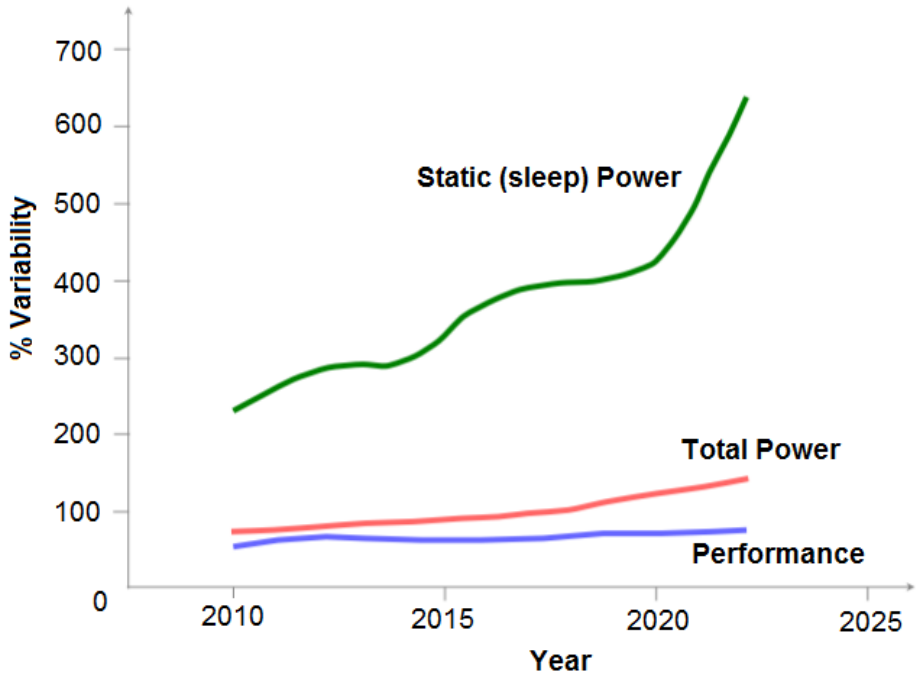

Figure 1.1. Circuit variability as predicted by the ITRS [22], showing that performance will stagnate even though variability in power consumption will continue to grow over the years.

just a rapidly growing market [39]. A majority of the world's population is using smartphones as the new general purpose computing machines, and a direct impact of the study of variability and of a fluid hardware-software stack on the lives of people can be more evident in this narrower setting. It is for this reason that this thesis seeks to build upon previous work in characterizing variability in embedded devices [42] [2] by extending the study to general purpose smartphone platform, to understand the nature of hardware variability in these processors, including how applications could take advantage of information about the presence of such variability. The expectation is that the same motivation that suggests the idea of a more robust, reliable and responsive machines also applies to mobile devices. The rest of this thesis focusses on smartphones as representative mobile devices that may be affected by process manufacturing variability.

1.2 Thesis Organization

This thesis is organized into the following sections:

Exploring the nature of processor power variability across mobile devices: Before we move on to software adaptability, it is important to verify that there is in fact measurable variability across devices, and characterize this variability with reasons for the observed differences. We observe a $\sim$6-15% variability in processor power in mobile devices.

Observing the effect of temperature on processor power variability across mobile devices: Temperature is usually a good indicator of power usage of a device and vice-versa. Thus, we seek to show that the cross-device variability of smartphones increases with an increase in ambient temperature. We observe an increase in processor power variability on mobile devices to $\sim$22-23% at significantly warmer ambient temperatures.

Making the case for adapting software to address mobile device processor power variability: Once we have established that there is significant variability across devices, we seek to demonstrate how a specific application could be adapted to use the information obtained by monitoring the device and how such parameter measurement-based decisions could themselves vary across devices, resulting in lower energy consumption across all devices. We use a canonical example of video playback, which is a common application on mobile devices to show how variability-aware adaptation can improve energy efficiency.

In the forthcoming chapters, we go on to illustrate our efforts in the exploration and characterization of variability in mobile devices (here, smartphones). Work related to the project we have undertaken is covered in Chapter 2. In Chapter 3 we discuss

the infrastructure, testbed and the experimental setup. The characterization of observed variability in the experiments is discussed in Chapter 4. In Chapter 5 we make the case for adapting software with variability-aware information with results of a proof-of-concept application. Lastly, Chapter 6 discusses the implications of the observed processor power variability across multiple smartphones and the adaptability of software to this variability, followed by a summarization of our results, as well as a discussion on future work.

Chapter 2

Related Work

Before we begin analyzing processor power variability across mobile devices, it is important to understand the reasons for variability and previous work in characterization of variability. In this chapter we discuss the work done along a few dimensions that intersect with our research, comprising of the push to expose hardware variability, creating an adaptive software stack and the extensive research to reduce the power consumption of smart phones from the perspective adapting applications for energy efficiency.

2.1 A fluid hardware-software interface

In some early work on exposing the hardware differences, Wang et. al. [41] advocate a common model to read hardware accelerator values through software. In a way, this creates a level of indirection over which applications interact with the hardware only through software. While not necessarily advocating exposing of the hardware variability, their case made for a more flexible hardware-software interface is endorsed by the vision of the variability expedition Gupta et. al. [16]. The authors also mention that hardware manufacturing conforming to rigid parametrized specifications is becoming expensive and a fluid hardware-software interface could lead to more robust systems. Gupta et. al. [16] envision Underdesigned and Opportunistic (UnO) computing machines that would provide a unified way of addressing variations due to manufacturing and ambient

operating conditions. These machines would be instrumented with inexpensive monitoring methods for hardware signatures to variation-aware operating system adaptation mechanisms.

Our research explores the UnO space, particularly with respect to smartphones. We expect that handling variability of hardware specification at run-time that would result in the selection of a different execution strategy in an UnO machine could result in power savings, and our work focusses on providing a proof of concept for such decision making for smartphone platforms.

2.2 Sources of Variability

Gupta et. al.[16] summarize the major sources of variability as follows:

Semiconductor Manufacturing Factors: Considerable literature has already focussed on the difficulties of conforming to rigid hardware specifications stemming from unpredictability of circuit designs [22]. The International Technology Roadmap for Semiconductors (ITRS) highlights power/performance variability and reliability management in the next decade as a red brick (i.e., a problem with no known solutions) for design of computing hardware. As an example, Dighe et. al. [10] show within-die performance variation of more than 25% at 0.8V in a recent experimental Intel processor. As already explained earlier, shrinking dimensions for chip design increase the variability and highlight a need for fine-grained control of these devices.

Environmental Factors: The environment in which a device operates can be affected by different environmental factors like humidity and ambient temperature. However, the ambient operating conditions of a device could also be defined by the location of a device and its usage scenarios. Each of these contribute to device variability. For example, in the automotive industry, operating temperatures vary from 30°C to 175°C resulting in

large power deviations [26].

Effects Due To Aging: Circuit aging, either through wires or transistors in integrated circuits suffering wear-out leads to system power and performance changes over time of usage. Some physical mechanisms eventually resulting in circuit aging include bias temperature instability (where threshold voltage of transistors degrade over time), hot carrier injection (where transistor threshold voltage degrades on switching), and electro-migration (where the wire width shrinks as more current passes through it).

Vendor Differences: Almost all hardware manufacturing currently involves multi-sourcing of parts with identical specification from different vendors. This is because single vendor sourcing is difficult for the scale at which hardware manufacturing operates. These multi-sourced components also lead to substantial differences in power and performance as shown in Figure 2.2.

2.3 Characterization of Variability

While the work in Section 2.1 is motivated partially by theoretical predictions of variability in the future, it is also important to analyze the nature of variability in devices currently being used empirically. Wanner et. al. [42] explain the observed variability in embedded sensors using multiple instances Atmel SAM3U microcontroller. Figure 2.1 shows the sleep power variability across temperature for five instances of an off-the-shelf ARM Cortex M3 processor. Wanner et. al. [42] observe a 14x variation in leakage power and a 10% variation in active power of the embedded sensors.

Similarly, Gottscho et. al. [15] observe variations of upto 12.29% for idle power within a single model of double dual rate third generation (DDR3) dual inline memory modules (DIMMS). Hanson et. al. [17] observe a 2x variability in active power across

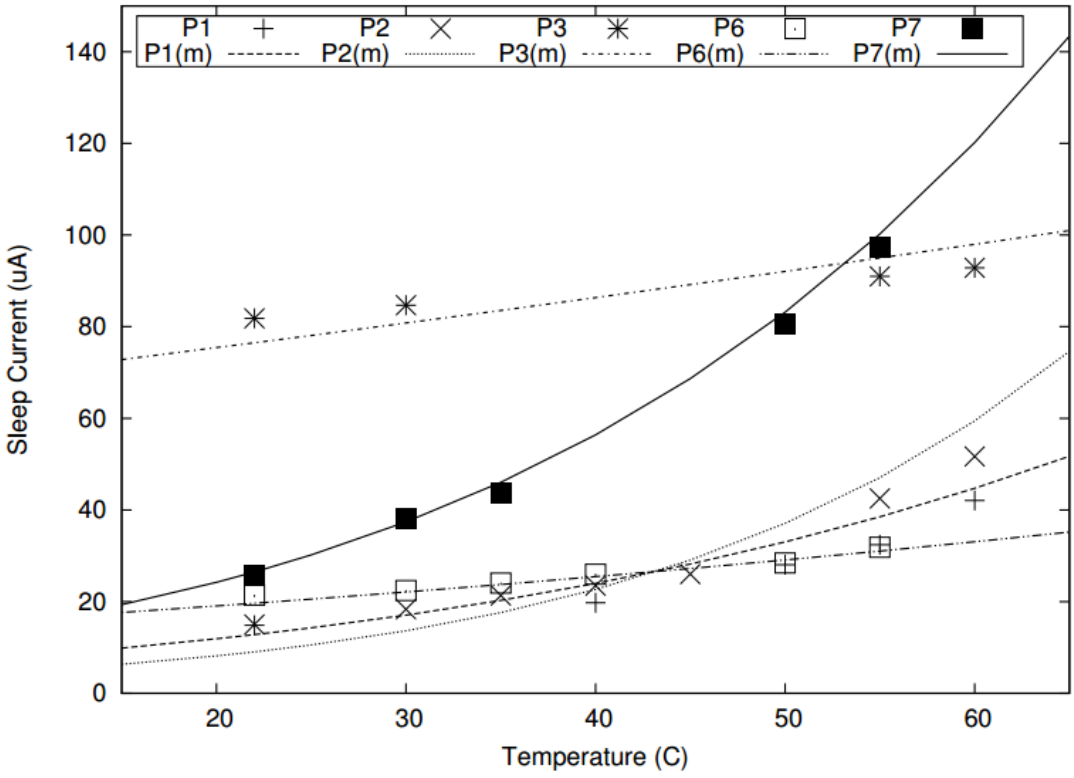

Figure 2.1. Sleep power variability across temperature for five instances of an ARM Cortex M3 processor [42].

identical DRAMS from different vendors, as shown in Figure 2.2.

In work that inspires our research Balaji et. al. [2] instrument Intel Core i5-540M laptop processors that are marketed in the same frequency bins, and thus presumed to be identical, and observes power variation ranging from 7% to 17% across different applications and configuration options, as shown in Figure 2.3. We also adopt a similar method of instrumenting smartphone platorms to measure the power consumed, focussing on characterizing the processor power variability across these devices.

2.4 Adapting for Variability

Writing software that reacts to hardware changes has always been one of the focal points of energy-saving techniques. For example, Choi et. al. [8] leverage spatial

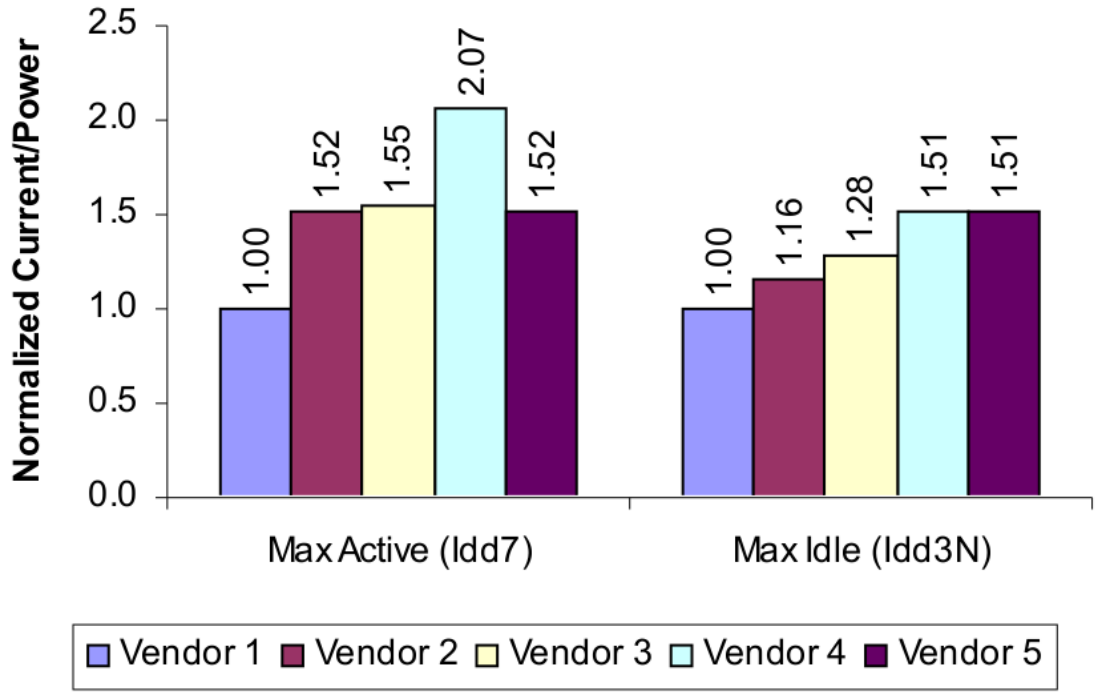

Figure 2.2. Power variability in DRAMS with identical hardware specifications manufactured by different vendors [17].

and temporal heat slacks to reduce on chip temperatures by dynamically changing the workload with operating system support. Thus, one of the most promising ways to reduce the negative effects of variability seems to be to write software that adapts to it. One such adaptation technique is to use the majority opinion while reading from hardware components as in Zhou et. al. [46]. This however, is not practical for general purpose computing machines where the goal seems to be to fit more into less space rather than provide redundancy in hardware to obtain accuracy. Thus, Pant et. al. [35] lay the ground for characterizing the hardware using hardware signatures to modify the algorithm used in the software layer accordingly. This resulted in a significant reduction in overdesign and an increase in overall quality of their observed application. One point to note here is that Pant et. al. [35] use video quality as their proof-of-concept application, which is also going to be the case in ensuing chapters of this work.

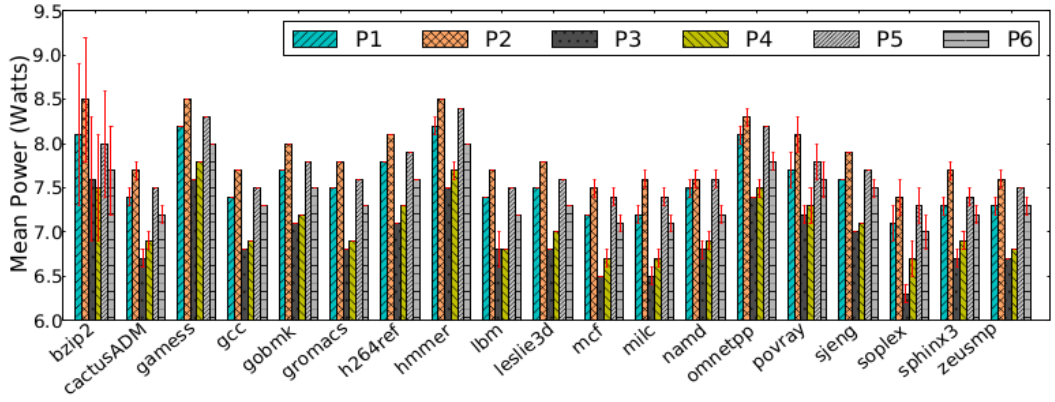

Figure 2.3. Power consumption of six Intel Core i5-540M processors for SPEC CPU 2006 benchmark 2.53GHz. Power variation ranges from 12% to 17% [2].

Wanner et. al. [43], similarly measure and characterize leakage power variability in current microprocessors and showed that variability-unaware estimates of power could leave upto 61% of the power untapped in long running embedded sensing systems and variability-aware duty cycling can lead to a 7.1x improvement in sensing quality for a fixed desired lifetime of their test sensor systems. Whereas many of these projects have been about the study of and software adaptation in smaller devices like sensors [43], it is clear that as feature sizes reduce for regular general purpose computing devices like the personal computer and the smartphone, the concerns over exposing hardware variability and writing software that can adapt to it will directly impact energy savings on these machines. A push in this direction comes from handling different hardware components of a general purpose computing machine and writing software to adapt to the variability in these components individually. Bathen et. al. [4] propose a hardware-assisted variability aware memory virtualization layer that allows programmers or applications to partition their address space into regions with different power, performance and fault-tolerance guarantees. Building on this work, Bathen et. al. [5] propose ViPZonE, a variability-aware software stack that allows developers to indicate to the OS the expected dominant usage patterns (write or read) as well as level of utilization (high, medium, or low)

through high-level APIs. Our research focusses on laying the foundation for similar work in software adaptation for smartphone devices, since it presents the next level of scale for embedded systems and such devices are becoming pervasive.

2.5 Energy Savings on Mobile Devices

A large volume of literature has been dedicated to energy savings on mobile devices given the express desire of users for longevity of battery lifetime. However, in this section, we only discuss the work that has been done in measuring, monitoring and conserving energy on mobile devices specifically from the perspective of addressing variability. We also explain the concepts that can be extended into our work on power variability in smartphones.

2.5.1 Instrumentation

An important goal of the variability expedition [1] has been to instrument and monitor power values and other parameters of the underlying hardware components that exhibit variation. This task is considerably harder in mobile devices which must meet the dual needs of users who wish to multi-task on their phones with long battery lives and manufacturers who would need extra sensors and power to monitor hardware parameters like frequency, temperature and power. In an earlier effort to improve battery lifetime of mobile devices, Flinn et. al. [13] combined hardware provided information with program modelling to produce an energy profile, much like the hardware signatures discussed in Pant et. al. [35]. In another approach, Oliver et. al. [33] presents an Energy Emulation Toolkit that allows application developers to evaluate the energy consumption requirements of their applications against real user energy traces. Oliver et. al. [33] also suggests the classification of users based on their charging patterns to determine what underlying algorithm an application must use. We suggest that a

similar classification be made for devices that would allow applications to choose what quality of service to provide to users based on some power consumption values or other sensor parameters. While this thesis only focusses on instrumentation to obtain fine-grained power measurements for the CPU subsystem, other platforms choose to address energy effiency in embedded systems by dynamically selecting the most energy efficient hardware components that meet specific requirements in sensing delity, computational load, storage media, and network bandwidth [30].

2.5.2 Adapting for Energy Efficiency

Once you have data about power or energy consumption and other parameters, one must go about adapting software on the phone to save energy consumed, and thereby extend battery lives of the devices. Paek et. al. [34] do this by determining when to turn on the GPS based on a pre-determined level of location accuracy that can be expected based on where the user is located. For example, in urban areas, GPS is generally less accurate, so it suffices to turn it on only to achieve that accuracy. Cheng et. al. [7] achieve about 14 to 20% energy savings with their proposed Quality Adaptive Backlight Scaling scheme that determines when to dim the backlight in devices equipped with a TFT (Thin Film Transistor) LCD (Liquid Crystal Display). The trade-off here is that a dimming backlight affects the brightness of a video and the backlight scaling should still meet some preset quality requirements. Mohapatra et. al. [31] optimize network usage of mobile devices and explore the trade-off between quality of service and energy savings. Lee et. al. [27] analyze the energy consumption of different video encodings in handheld devices. Martins et. al. [29] allow each individual user to determine what quality they would like for an application like audio/video or navigation and present energy profiles for the different configurations available for these applications. For example, for their navigation application, they present the user with options ranging from

the least power-consuming choice of displaying only a list of directions from source to destination to the highest power-consuming choice of enabling real-time turn-by-turn navigation with voice instructions.

In work well ahead of its time, one pioneering system for adapting applications for energy efficiency on mobile devices was demonstrated by Flinn et. al. [12] in their Odyssey system where they modify the Linux operating system to predict future energy demands from measurements of past usage, and notifying applications to adapt when there is substantial mismatch between predicted demand and available energy. Their system extended battery life by around 30%.

As is rather evident from many of these adaptive applications, they all leverage some configuration options of the application and tune some parameter to provide appropriate quality of service to the user as well as longer battery lives by saving energy consumed. In our work, we make the case for adaptive software by using video playback as our proof-of-concept application, as has been adopted by [7] [13] [29] [31] since video playback has a measurable, agreed-upon quality metric as well as tunable configuration parameters that will be discussed in Chapter 5. We suggest that variability in smartphones be dealt with as yet another tunable parameter by which we can beget considerable energy savings by making wise decisions about the state of our software applications.

Chapter 3

Experimental Infrastructure

An important aspect of experimental research is the infrastructural support required to be setup before conducting the experiment. With respect to our work, before we make observations about processor power variability in mobile devices, we need to explain the infrastructure we used to obtain our measurements and the experimental workflow. In this chapter, we describe the instrumentation setup we used for our experiments to explore the nature of mobile device variability to observe and record power measurements for individual system components.[1]

3.1 Measurement System

The basic infrastructure for our analysis is provided by Qualcomm, consisting of the hardware for instrumentation and the software to obtain data from the instrumented device. A complete overview of the setup is shown in Figure 3.1, and is described below.

[1] Some of the content described in this chapter may seem abstract or superficial since specific details are covered under a Non-Disclosure Agreement with Qualcomm. If you can request access to that information under a similar agreement, then you may refer to the WQEPM User Guide [38] for more information on the WQEPM application.

3.1.1 Hardware

For our analysis on smartphones, Qualcomm has provided five devices belonging to the Snapdragon MSM8660 Mobile Development Platform (MDP). Each device runs the Android Operating System (Ice Cream Sandwich, Android 4.0.3). The details about these devices are listed in Table 3.1. These devices are labelled Dev1-5 in future references.

Table 3.1. Device information for the Snapdragon MSM8660 Mobile Development Platform (MDP). [37]

Processor	MSM8660 with asynchronous dual-core central processing unit (CPU) cores at 1.5GHz each
Graphics	Adreno 220 graphics processing unit (GPU)
Display	3.61" WVGA capacitive multi touch screen
Video	1080 high-definition video recording and playback up to 30 frames per second Stereoscopic 3D playback via HDMI output
Camera/Camcorder	13 megapixel main camera w/ LED Flash 1 megapixel front camera
Audio	Dolby 5.1 audio
Memory	1GB LPDDR2 RAM 16GB on-board flash External SD slot with 8GB SDHC card included
Connectivity	802.11 a/b/g/n Wi-Fi, Bluetooth, GPS, FM
Keys	Dual stage camera shutter with half press Volume/zoom +/- switches (context dependent) Power on/off key HW reset (recessed) OS-specific soft keys
Connectors	USB OTG micro connector with USB charging HDMI type D connector 3.5mm audio jack Micro SD external slot

Qualcomm also provided a National River Technologies (NRT) MiniBoard that allows us to monitor and measure the current and/or power supplied to specific system components in the smartphones. This NRT MiniBoard shall henceforth be referred to as the Debug Board. As shown in Figure 3.1, the Debug Board acts as the interface between

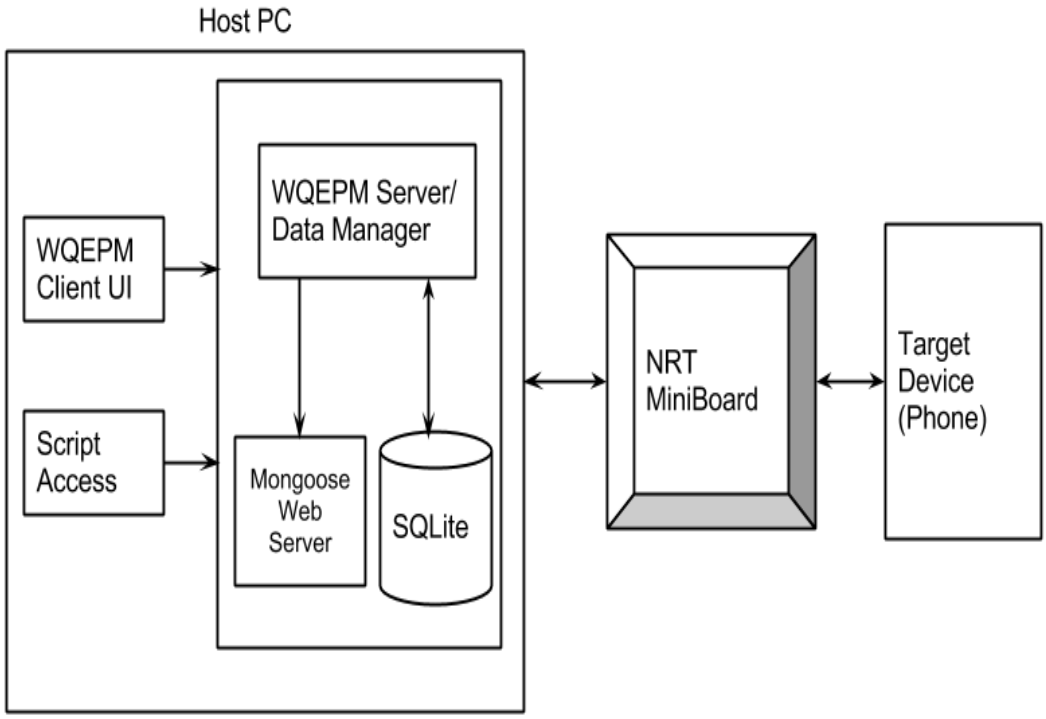

Figure 3.1. Infrastructure diagram for the setup used to monitor and record power measurements for each phone. [38]

our MSM8660 device and the software to extract measurement data from the device. Not shown in the figure, however is the actual layout. The Debug Board has a specific slot for the phone and the connector acts as the battery emulator for the phone. The Debug Board is connected to a host PC that runs the monitoring software via a USB connector. The phone is also directly connected to the host PC so as to use the Android Debug Bridge (ADB) [20] to run scripts and applications on the smartphone.

3.1.2 Software

A large part of the setup shown in Figure 3.1 resides on the host PC, typically a Windows machine. Qualcomm has provided an application called the Web-based Qualcomm Embedded Power Monitor (WQEPM), which allows us to monitor power channels on the target device, here the smartphone, via scripts running on an independent host PC, so that we have no overhead of measurement on the device itself. This application when launched connects to the Debug Board and monitors the target device. To obtain any information from this device one needs to use a browser based client UI or write a client script to specify the components to be measured. WQEPM writes all its values to an SQLite database from which our scripts can then extract information.

Figure 3.2 shows a screenshot of the WQEPM Web User Interface to interact with the Debug Board. The legend on the right indicates the power measurement points or channels being monitored. While this UI allows a user of the WQEPM software to view the data being extracted from the Debug Board graphically, it also allows the user to export this data once the monitoring has been stopped as a CSV file, which supports extensive scripting. The window on top of the browser is the WQEPM monitoring setup that connects the Debug Board with target information. For our experiments however, the process to get information from the Debug Board is handled directly by scripting access to the SQLite database, and the graph visualization is not used to observe device power variability.

Another important software component we use is the Android Debug Bridge (ADB) [20]. ADB allows us to connect directly to the smartphone and automate the running of applications on the phone without needing a user to interfere and start/stop an application. ADB also provides a Linux-based shell to read, update or modify phone parameters such as maximum and minimum operating frequency, which frequency

governor to use, what cores to use, etc. which we will use extensively in our experiments as described in Section 3.3 subsequently.

3.2 Measurement Parameters

The instrumentation provided by the Debug Board allows us to monitor and record power and/or current supply values for 32 different power measurement points or channels. The broad categories for these channels are Audio, Camera, Core, Display, Memory and some miscellaneous channels. A comprehensive list of these power measurement points is described in Table 3.2. While it is tempting to be able to draw fine-grained power measurements from all the specified channels, our goal to explore power variability in processors across mobile devices does not require such measurements, nor do the applications we use stress all these channels.

For the purposes of exploring processor power variability, of the various power measurement points or channels, we focus on one particular channel for Internal Memory and three different Core channels – the two Scorpion processor Cores on the device and a Digital Core which handles peripheral components of the CPU subsystem[2]. The reason for this choice is two-fold. First, we focus on device power variability based on a subset of system components. For our project, we only focus on processor core variability. For our desired granularity of components, we shortlisted around eight channels as prescribed by correspondence with Qualcomm. Second, even the subset of power measurement points we considered for our focus of system components can be narrowed down further based on the actual contribution of those various channels to the total power usage of that subsystem. In our case after we considered 8 channels or power measurement points that were likely to impact the CPU subsystem, we then dropped the ones that made no

[2]This information was obtained from personal correspondence with Qualcomm. For more information on the channels measured, please see Appendeix A.1.

Figure 3.2. This figure shows a screenshot of the WQEPM UI to interact with the Debug Board. The legend on the right indicates the power measurement points or channels being monitored. While this UI allows a user of the WQEPM software to view the data being extracted from the Debug Board graphically, it also allows the user to export this data once the monitoring has been stopped. The window on top of the browser is the WQEPM Monitoring setup that connects the Debug Board with target information.


WQEPM
File View Targets Command Help
QUALCOMM
WQEPM
WQEPM
WQEPM IP :7376/wqepm/?s= WQEPM IP :7376
An A-Z Index of the ...
WQEPM
QUALCOMM
WQEPM IP Address
MSM8660MDP
New Chart
Start
Stop
Paused
Share
Export
Settings
Help
Chart 1
Select Channels
VREG_S0A mA
VREG_S0B mA
VREG_S1A mA
VREG_S1B mA
200.00
100.00
0.00
16:11:08.333
16:11:09.167
16:11:10.000
16:11:10.833
16:11:11.667
16:11:12.500
Auto scaling
Goto Time
- 25s
- 5s
- 2.5s
+ 2.5s
+ 5s
+ 25s
Zoom
5,000
msecs
window
STOPPED
WQEPM Server Time:
Time on Host PC

or minimal contribution signified by a constant low power value collected and reported by WQEPM. The granularity and density of power measurements we obtain from the different channels gives us a detailed view of the processor power variability across mobile devices, which will be discussed in Chapter 4.

Table 3.2. This table provides a comprehensive list of all the power measurement points available on the MSM8660 MDP. [38]

Category	Power Rail Name	Description of Functions Covered By Power Measurement Point
Audio		
	VREG_S3B	Audio DSP
	VDDD_CDC_IO	Audio Codec I/O
	CDC_VDDC	Codec Analog
	CDC_VDDA	Codec Digital (VREG_L5)
Camera		
	VREG_LVS0A	Camera I/O
	VREG_L25A	Camera (digital)
	VREG_L15A	Camera (Analog)
Core		
	VREG_S0B	Scorpion Core 0
	VREG_S1B	Scorpion Core 1
	VREG_S1A	MSM8660 Digital Core
Display		
	OLED_ELVDD	Active-Matrix Organic Light-Emitting Diode (AMOLED) Electroluminescence Virtual Fourier Filter
	OLED_VDD3	AMOLED I/O
	OLED_VCI	AMOLED Memory
I/O		
	VREG_L5A_TS	Touchscreen
	VREG_S3A_PX3	MSM I/O Pad 3
	VREG_L5A_PX2	MSM8660 I/O Pad 2
	VREG_L4B	Haptics
Memory		
	VREG_L14A	Micro SD
	VREG_S0A	MSM8660 Internal Memory
	VREG_L5B	eMMC power supply line for internal flash

Table 3.2. – continued

Category	Power Rail Name	Description of Functions Covered By Power Measurement Point
	VDD2_ISM	MSM8660 Internal Stacked Module (Voltage Domain 2)
	VDDPX1_LPDDR2	DRAM - input receiver power supply, I/O power supply, MSM power for I/O pad group 1 - External Bus Interface and System Management Interrupt
Misc		
	VDD1_LPDDR2	DRAM Voltage Domain 1
	VREG_LVS0B	eMMC power supply line for host interface
	VDD1_ISM	MSM8660 Internal Stacked Module (Voltage Domain 1)
	VDD2_LPDDr2	DRAM Voltage Domain 2
	VREG_L16A	Phase Lock Loops, HDMI, Camera Serial Interface 2/4
Modem		
	VREG_L13A	COMBO_DAC, SVIDEO, BBRX
Sensor		
	ALS_VDDA	Ambient Light Sensor
Total Power		
	VPH_PWR	Main Power Supply (3.3 - 4.2V)
Video		
	VREG_HDMI_5V	HDMI_HPD, HDMI_LVL

3.3 Experiment Workflow

Before we explain the actual experiment, it is important to establish our experiment workflow using the infrastructure just described in Section 3.1 above. A typical experiment follows this procedure:

For each device:

1. Stop *mpdecision* process.

2. Turn off core 1 of the two Scorpion cores, labelled Core0 and Core1.

3. Set CPU frequency governor to *userspace*.

4. Set CPU frequency to some specific frequency (usually between 192MHz and 1.5GHz).

5. Start WQEPM and connect to the required target.

6. Run script (in Perl) for experiment.

7. Stop WQEPM to unlock the database.

8. Run script to extract data collected.

For our experiments to measure variability, we must be able to record processor power by keeping as many factors constant as possible. That is the reasoning behind Steps 2 to 4. Some applications may use both cores, while some may not. To test variability across a single core, we restrict our applications to use only Core0 and thus turn off Core1 by stopping the *mpdecision* process that handles frequency scaling and core allocation. The MSM8660 devices come programmed to run at specific frequencies ranging between 192MHz and 1.5GHz. Setting them at one specific frequency allows us to assert that we are analyzing variability that is not induced by each device running at its own frequency and Step 1 and 3 allow us to turn off dynamic voltage frequency scaling. Steps 1 to 4 can all be achieved through ADB as mentioned in Section 3.1.2[3]. The process for Step 5 is described in the WQEPM User Guide [38]. Step 5 usually involves running the application that we would like to obtain power measurements for over the MSM8660 MDP. In order to extract the data measured from the SQLite database, we need to ensure that no other process is holding the database lock and therefore, we need to stop the

[3]Some of the actual ADB commands used are shown in Appendix B.2

WQEPM monitor which locks the database when monitoring channels even if it is not writing to the database.

When we have collected data across all devices, we can then analyze them to characterize variability across devices. Unless otherwise specified, all our future experiments undergo 10 runs on each device, and the static supply voltage for each channel or power measurement point is 1.1V.

3.3.1 Eliminating Measurement Errors

We have taken special care to ensure that the variation in values observed does not manifest due to measurement errors. We perform multiple runs of the same experiments on all the devices. Each experiment runs ten times on each device. All the devices start in the same state. We reboot the devices and wait for two minutes until after reboot to start the experiment, when the ADB shell *top* command always has the *top* process itself listed as the highest user of the CPU. Between runs, the device is given a sleep period ranging from 30 to 40s, so that intermediate setup and teardown does not interfere with application power measurements. These additional experimental runs are included in our results and are part of the standard deviation.

For the results that will be discussed in Chapter 4 for CPU-intensive benchmarks, the standard deviation is always less than $\pm 2\%$ of the total power at normal temperatures and less than $\pm$ 4% at warmer temperatures (around 50$\circ$C). Therefore, for a confidence interval of 95%, our margin of errors for a particular device are within $\pm 1.2\%$ of the total power at normal temperatures and within $\pm 2.4\%$ of the total power at warmer temperatures. The highest observed standard deviation across individual channels (here, Internal Memory) is around $\pm 6\%$ and the margin of error for this is within $\pm 3.7\%$ of the power measured for that channel. For more details on the calculation of these margins of error, please refer to the Appendix A.4. As our results in Chapter 4 will show, the

standard deviation is not high enough to be interfering with our results for measured variation.

Chapter 4

Processor Power Variability in Mobile Devices

With our experimental infrastructure in place, we now explain the benchmark applications we used to measure and monitor the processor power or CPU subsystem power variability across mobile devices. We characterize the observed variability, including a discussion on the potential reasons for our observations. We also present results for the effect of temperature on the observed device processor power variability.

4.1 CPU Subsystem Variability

4.1.1 Applications

For our first experiment, we consider CPU-intensive application benchmarks. Since the devices run on the Android operating system, applications for some of these benchmarks are already available to us [28]. While some of these applications can be used to measure the performance of a device in their own way (floating point operations per second(FLOPS), etc.), we only use these applications because they are known to stress the CPU subsystem. Unless otherwise specified, the devices are labelled Dev1-5. We are interested in the power consumed while running these benchmarks and not particularly about the performance results. The applications we consider for this experiment are:

1. *LINPACK*: In its most basic form, this standard CPU benchmark measures the number of floating point operations per second to see how fast a computer solves a dense n x n system of linear equations $Ax = b$ [11].

2. *Whetstone*: This is also another standard CPU performance benchmark mainly to measure floating point arithmetic performance [9].

3. *Dhrystone*: This is similar to Whetstone except that it is designed to measure integer arithmetic performance over a typical mix of applications that can be contained in small memory subsystems [44].

4.1.2 Observed Variability

As shown in Table 4.1 and Table 4.3, we observe about 15% maximum variability across devices for processor power consumption when the devices run at 192MHz and between 6% to 10% maximum variability when the devices run at 1.5GHz, where the two frequencies are two ends of the available frequency spectrum for these devices. Thus, there is observed variability ranging from 6% to 15% across our test smartphones for these CPU-intensive applications. Assuming an average battery lifetime of 8 hours[1] for a smartphone [18], this provides the scope for power savings ranging from half an hour to 70 minutes.

An overview of the variability observed across the devices when the CPU-intensive applications are run at 192MHz and 1.5GHz is shown in Table 4.1 and Table 4.3 respectively. A more elaborate view of the processor power variability when the devices run CPU-intensive applications along with a breakdown along three channels – Scorpion Core0, the Digital Core and Internal Memory for devices at 1.5GHz is shown in Figures 4.1, 4.2 and 4.3. Figure 4.1, Figure 4.2, Figure 4.3 show the absolute and normalized

[1] Assuming continuous phone usage, the iPhone5 is guaranteed to last for 8 hours on 3G.

Table 4.1. A tabular representation of the maximum and minimum power consumption across devices for CPU-intensive applications at 192MHz. The table shows the total variation observed across the devices for these specific applications as a percentage of the least power consumption.

Application	Maximum Power Consumption (mW)	Minimum Power Consumption (mW)	Maximum Device Variability
LINPACK	225.77	195.15	15.69%
Whetstone	235.61	203.75	15.64%
Dhrystone	226.35	195.40	15.84%

power usage when the devices run the LINPACK, Whetstone and Dhrystone benchmarks respectively. The graphs on the left show the absolute values of power supplied to the three channels in mW, while the graphs on the right show the same values normalized to the first device. The normalized values show us the variability across devices as a percentage of power consumed by one device (always set to Dev1). Dev4, depicted by the straight line hatch, always uses the least power, whereas Dev5 almost always consumes the highest total power due to the highest power contributions from its core0 and internal memory. From the normalized graphs, we also observe that the Internal Memory and Core0 seem to supplement each other's contributions to overall device variability, i.e., the ordering of devices based on the power variability is the same for Core0 and Internal Memory. The digital core, on the other hand, complements them, as the ordering of devices by Digital Core power is different from that of Core0 or Internal Memory. This is also asserted by Table 4.2 and Table 4.4, which shows that the power variability across the devices on a per-channel level are much higher than the overall processor power variability, since the the devices with the minimum and maximum power vary across the power measurement points.

To better understand this discrepancy in channel contribution, we present another view of the total power variability for CPU-intensive applications in Figure 4.4. Figure

Table 4.2. The percentage of power variability on a per-channel level for three channels: Core0, Digital Core and Internal Memory when the devices ran at 192MHz. These values are significantly higher than overall power variability because the ordering of the highest and lowest power devices does not remain the same across channels.

Application	Scorpion Core 0	Digital Core	Internal Memory
LINPACK	24.75%	20.23%	47.48%
Whetstone	17.59%	21.34%	44.89%
Dhrystone	28.75%	20.00%	46.46%

Table 4.3. A tabular representation of the maximum and minimum power consumption across devices for CPU-intensive applications at 1.5GHz. The table shows the total variation observed across the devices for these specific applications as a percentage of the least power consumption.

Application	Maximum Power Consumption (mW)	Minimum Power Consumption (mW)	Maximum Device Variability
LINPACK	742.44	675.78	10.03%
Whetstone	881.70	827.83	6.51%
Dhrystone	650.10	588.44	10.48%

Table 4.4. The percentage of power variability on a per-channel level for three channels: Core0, Digital Core and Internal Memory when the devices ran at 1.5GHz. These values are significantly higher than overall power variability because the ordering of the highest and lowest power devices does not remain the same across channels.

Application	Scorpion Core 0	Digital Core	Internal Memory
LINPACK	14.90%	9.70%	44.48%
Whetstone	8.13%	9.92%	30.18%
Dhrystone	21.46%	9.56%	40.22%

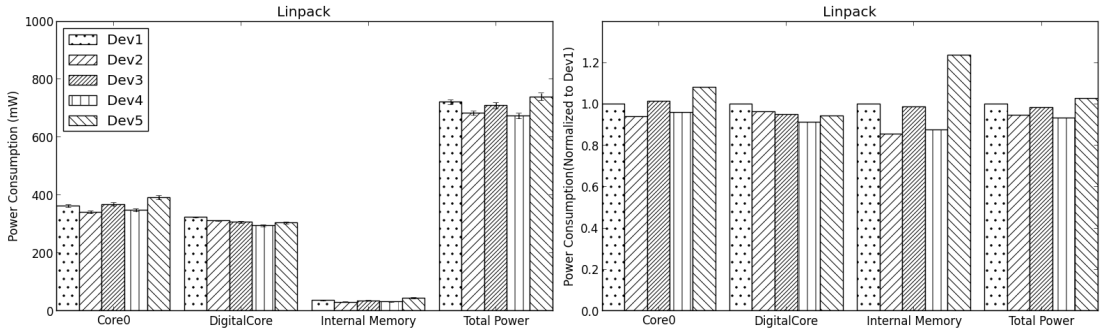

Figure 4.1. Power usage across devices for LINPACK, one of the CPU-intensive applications, showing power used by three channels: Core0, the Digital Core and the Internal Memory. The last column shows the total processor power. The two graphs show absolute and normalized power respectively. The devices were running at 1.5GHz.

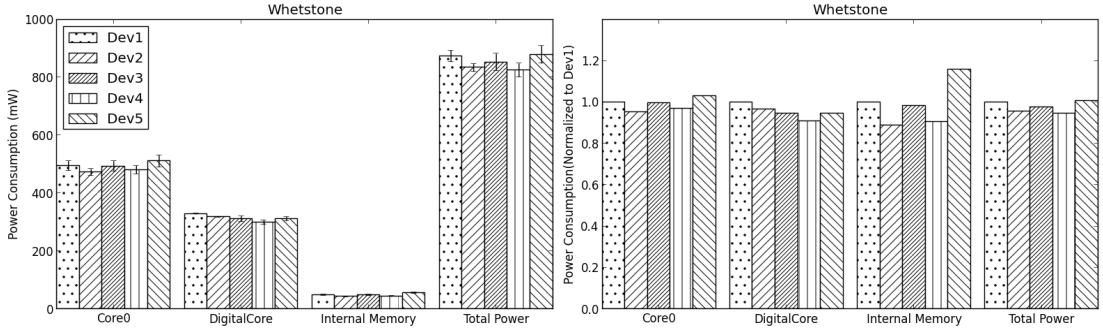

Figure 4.2. Power usage across devices for Whetstone, one of the CPU-intensive applications, showing power used by three channels: Core0, the Digital Core and the Internal Memory. The last column shows the total processor power. The two graphs show absolute and normalized power respectively. The devices were running at 1.5GHz.

4.4 (a) shows the total power for all devices as a sum of the contributions of the power to the various channels normalized to a percentage contribution of the total processor power. This shows us that the contributions from the various channels does not differ by a very significant amount and thus, we may be able to express these contributions as a fraction of the total power. It can be observed from Figure 4.4 (a) that, for each application, it is possible to express the total power, P_{total}, as an equation with constant contributing factors:

$$P_{total} = P_{core0} + P_{DigitalCore} + P_{InternalMemory}$$

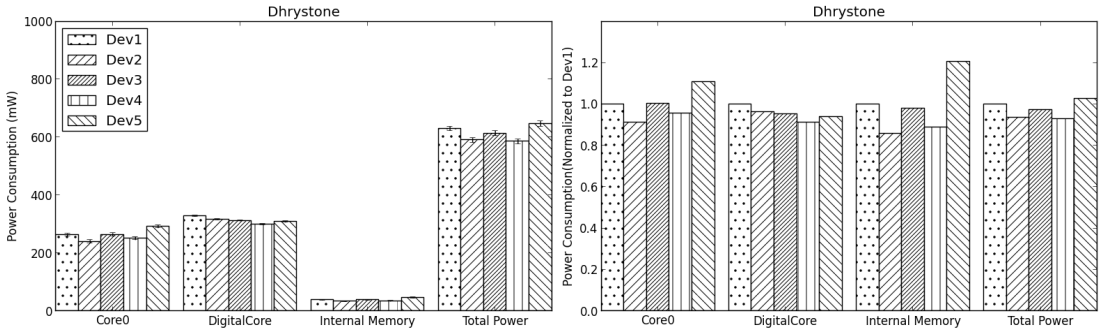

Figure 4.3. Power usage across devices for Dhrystone, one of the CPU-intensive applications, showing power used by three channels: Core0, the Digital Core and the Internal Memory. The last column shows the total processor power. The two graphs show absolute and normalized power respectively. The devices were running at 1.5GHz.

$$P_{total} = C_1 * P_{total} + C_2 * P_{total} + C_3 * P_{total}$$

where C_1, C_2, C_3 are constants that determine the contribution to overall device variability as the values for P_{core0}, $P_{DigitalCore}$ and $P_{InternalMemory}$ vary by device. This endorses our argument about device variability in smartphone processors being amplified by inherent system component variability.

Figure 4.4 (b) shows the absolute contribution of the three power measurement points to the total power. The variability across the devices, even at the per-channel level is quite clear even though the relative contributions to the total power for these different channels remains almost the same. As explained earlier through Tables 4.2 and 4.4, the per channel variability is actually higher than the overall device variability, since in many devices, when the Core0 and Internal Memory contribute towards increasing the processor power variability across the devices whereas the digital core contributes to variability in the reverse direction.

4.2 Impact of Ambient Temperature

As mentioned in Section 2.2 and by Wanner et. al. [42] in their work on embedded devices, device variability can be amplified by ambient temperature. In this section we

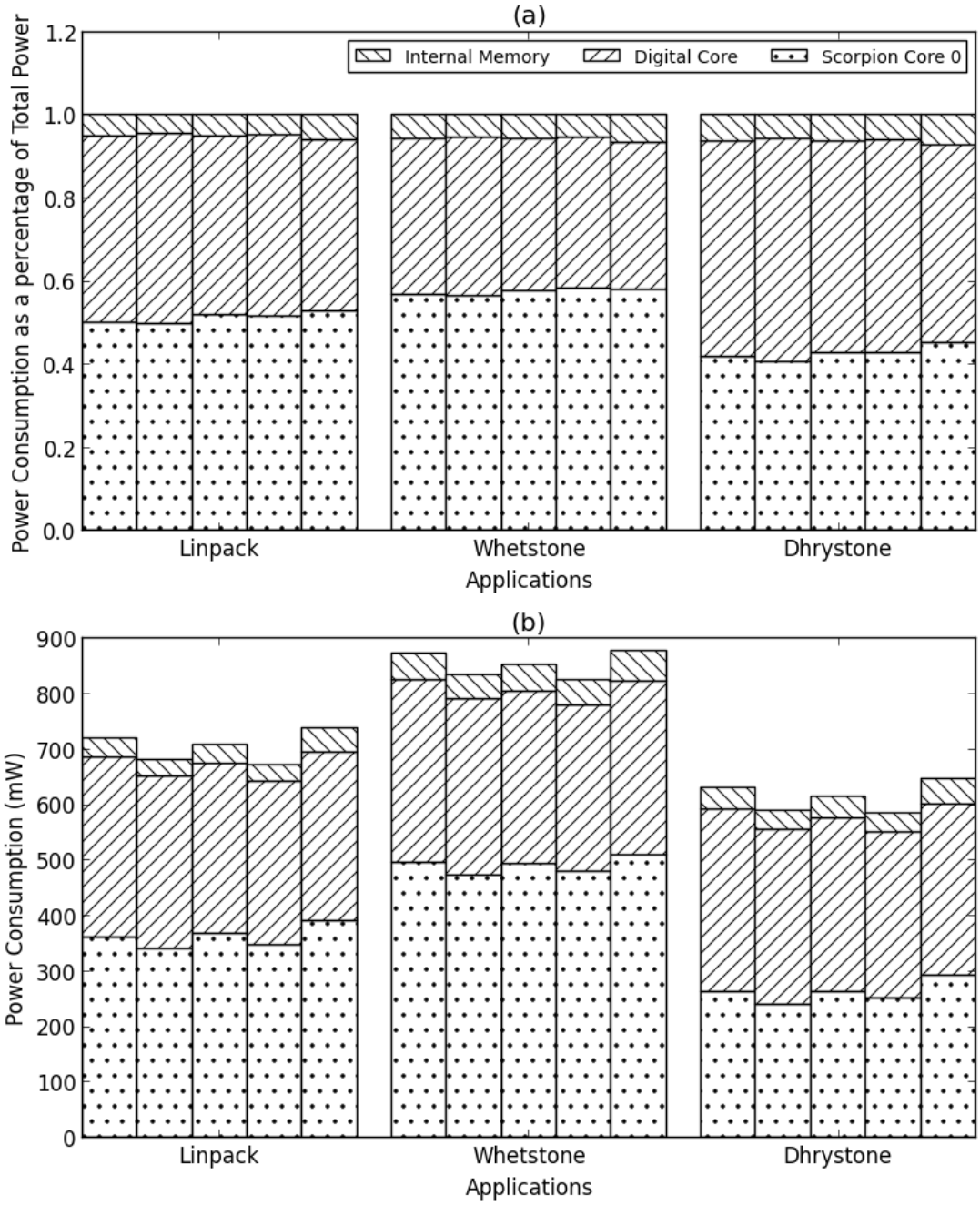

Figure 4.4. (a) Shows the relative contributions of the measured channels to the total power, on a scale normalized to 1 to indicate fractional contributions. (b) Shows the variability across the devices in terms of total processor power with absolute values of contributions from the three channels: Core0, Digital Core and Internal Memory. The devices were running at 1.5GHz. In both figures, each bar in the group of 5 represents a test device, Dev1-5 in that order.

explore whether this really is the case in mobile devices with respect to processor variability. We consider the CPU-intensive applications discussed in Section 4.1.1 as well as Video Playback, the details for which will be covered in Chapter 5.

The workflow is as discussed in Section 3.3, with one minor addition: when testing warmer ambient temperatures, we use a space heater [2] to raise the ambient temperature significantly more (between 50°C and 55°C) than the normal temperatures (room temperatures between 27°C and 33°C)[3]. We choose these significantly different temperature ranges for two reasons. First, our experiments often run for half an hour on a single device. Continuously operating these devices with applications that stress the CPU subsystem will heat up the phone by a few degrees as we near the end of our experiment [4]. Second, small changes in temperature will not necessarily show any differences in variability since hardware devices are designed to handle these small variations and we would be unable to distinguish between variability caused by changes in temperature and variability observed through experimental error.

Figure 4.5 shows two sets of values for each device. The bottom set shows the total processor power across devices per application (Linpack, Whetstone and Dhrystone) at normal temperature ranges of ∼27°C to 33°C. The top set shows the total processor power across devices for the same applications at significantly warmer temperatures of ∼50°C to 55°C. As shown in Table 4.5, the device processor power variability is amplified from a range of ∼6-10% to 14-17% for the CPU-intensive benchmarks.

Similarly, Figure 4.6 shows that the variability across devices is significantly amplified by warmer ambient conditions for a video playback application as well. Without discussing the details of the actual application run (which will be explained shortly in

[2] We thank Jennifer Folkestad and CNS for the heater.

[3] Since we do not use sophisticated equipment to maintain temperature, we keep our temperature ranges significantly different to analyze changes in device processor power variability.

[4] We observe that the power increases slightly for run to run variation because of this temperature change in Section A.2

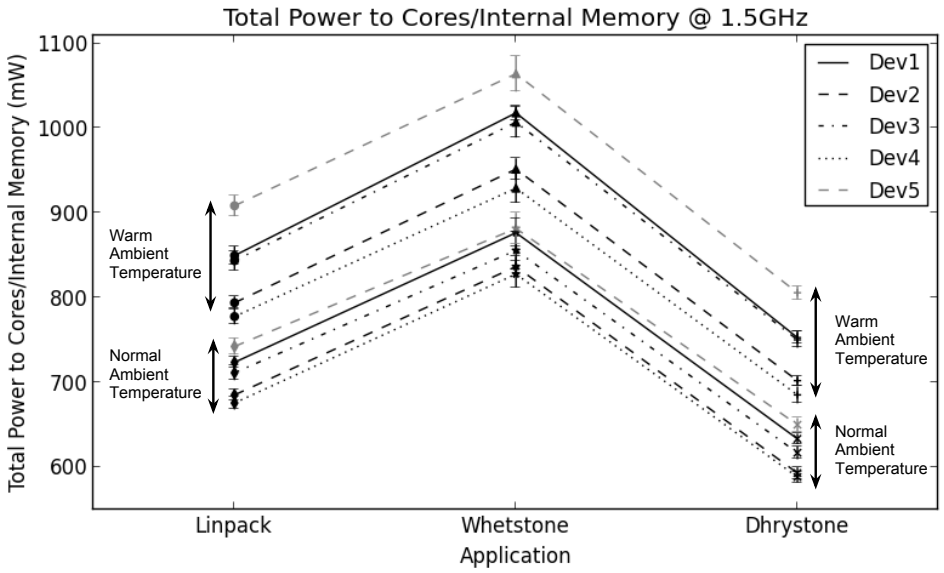

Figure 4.5. This figure shows the effect of increased ambient temperatures on processor power variability when the phones run cpu-intensive applications like Linpack, Whetstone and Dhrystone at 1.5GHz. We observe that the variability across devices is amplified at warmer ambient temperature ranges of 50°C to 55°C compared to the variability observed at normal temperature ranges of 27°C to 33°C.

Chapter 5), it is evident that processor power variability across the devices is amplified from around 13% at normal temperatures to ∼22-23% at warmer temperatures, as summarized in Table 4.6.

4.3 Discussion

In this section, we provide explanations for the observed variations from the previous sections, and discuss a few points of importance when understanding the results. In Figure 4.4, we observed that the per-channel power variability across devices differ by channel. Table 4.2 echoes this result, and shows the per channel variability across devices for the different CPU-intensive benchmark applications. We observe that the values for per-channel power variability across devices are significantly higher than the overall

Table 4.5. This table shows the processor power variability across the devices when they run CPU-intensive benchmarks (like LINPACK, Whetstone and Dhrystone) at a normal temperature range of ∼27°C to 33°C and a warmer temperature range of ∼50°C to 55°C.

Application	Maximum Device Variability (Normal Temperature)	Maximum Device Variability (Warm Temperature)
LINPACK	10.03%	16.81%
Whetstone	6.51%	14.52%
Dhrystone	10.48%	17.58%

Table 4.6. This table shows the processor power variability across the devices when they play a video with three different encoding formats (3gp, H.264 and Webm) at a normal temperature range of ∼27°C to 33°C and a warmer temperature range of ∼50°C and 55°C.

Video Encoding Format	Maximum Device Variability (Normal Temperature)	Maximum Device Variability (Warm Temperature)
3gp	13.31%	23.14%
H.264	13.64%	23.64%
Webm	13.58%	22.73%

processor power variability observed in Table 4.1. The same observation can be made from Table 4.4 as compared to Table 4.3 for the devices running at 1.5GHz. We believe that the reason for the larger differences at a per-channel level not being manifested in overall processor power variability is because the devices that consume the highest and lowest power at a per-channel level are not the same. For example, for Core0 at 1.5GHz, Dev5 is the highest power device and Dev2 is the lowest power device. However, for the DigitalCore, Dev1 is the highest power device and Dev4 is the lowest power device. While Core0 and the Digital Core complement each other's power contributions, the combined effect of the high values for Core0 and Internal Memory always render Dev5 as the high power device similarly render Dev4 the lowest power device.

We also observe that the variability at 1.5GHz is actually less than that at 192MHz.

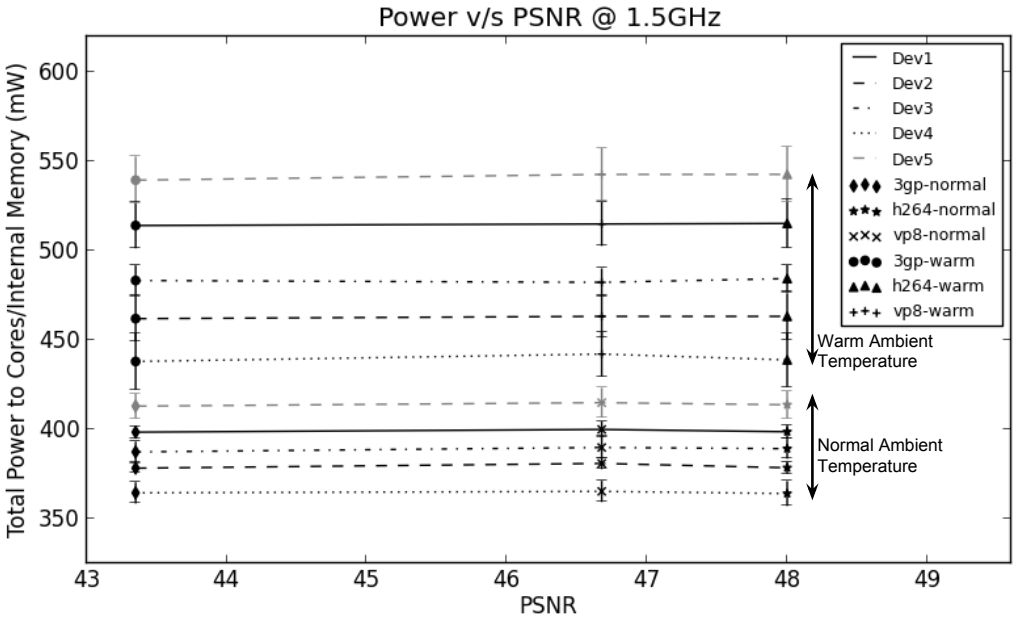

Figure 4.6. This figure shows the effect of increased ambient temperatures on processor power variability when the phones play three different encodings of the same video (3gp, H.264 and Webm) at the same frame resolution at 1.5GHz. We observe that the variability across devices is amplified at warmer ambient temperatures.

This could be because all the devices consume far less power at 192MHz and therefore, any small changes across devices are amplified when expressed as a percentage of the original lower power. At 1.5GHz, all the devices operate at a significantly higher power, even in idle as shown in Figure 4.7. Another reason for this could be that the dominance of leakage power is more prominent at 192MHz at which the power consumed by the system components is definitely lower than that at 1.5GHz. In our experiments comparing the devices at different frequencies, we have only scaled the frequency and the voltage to the components is assumed to be the same since the configuration file does not specify different voltage values at different frequencies.

We observe that there is significant variability even when there are no specific applications running on the devices and the phones are in an idle state. Figure 4.7 shows

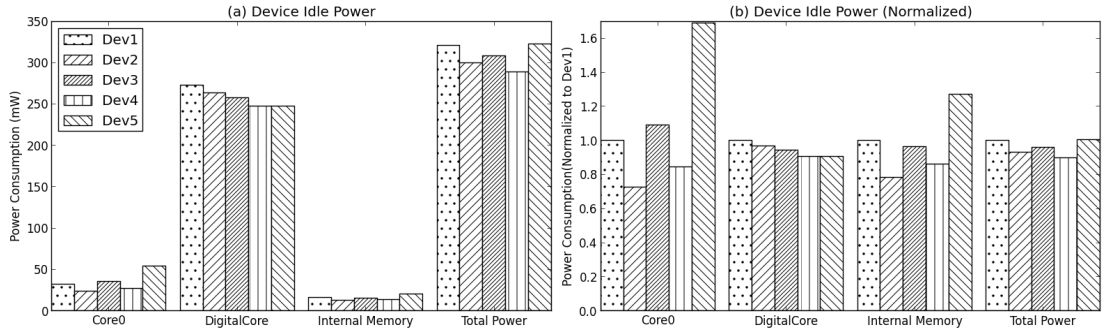

Figure 4.7. The graphs show the absolute and normalized idle power usage across devices for three channels: Core0, the Digital Core and the Internal Memory, and the total processor power, with the devices running at 1.5GHz.

the absolute and normalized idle power usage across devices for three channels: Core0, the Digital Core and the Internal Memory, and the total processor power, with the devices running at 1.5GHz. Figure 4.7 (a) has absolute values and Figure 4.7 (b) has values normalized to Dev1. The figure also shows the idle-power values for the three channels: Core0, the Digital Core and the Internal Memory. We observe in Figure 4.7, the idle channel power is a significant contributor to the overall processor power variability. In an idle state, the processor power across the devices varies by about 11.57%. The processor power variability actually decreases when we run applications for a similar reason as the higher variability observed at 192MHz. The values of idle processor power are significantly smaller than the values of processor power when the devices are in operation.

Another observation to be made is that since we were running CPU-intensive benchmark applications, Core0 is the major contributor to the total power of the devices. We will observe in Chapter 5 when we discuss video playback that the digital core, which is responsible for peripheral components of the CPU subsystem, becomes a major contributor to the total power, while also increasing the variability observed.

We know that power is proportional to temperature, so it is expected that a device

requires more power to operate at higher temperatures. However, we observe that this increase in the power consumed at higher temperature is also different across the five devices. As mentioned earlier, this could be because the contribution to total power from the leakage power is a dominant factor at higher temperatures and is responsible for the increased variability. We observe larger differences across the processor power at warmer temperature ranges of $\sim$50°C to 55°C, compared to the differences observed at normal temparture ranges of $\sim$27°C to 33°C. The increase in variability at higher temperatures could also be because the variability across the idle operating power of the devices also increases, and as we have previously noted, it is a contributor to the overall variability of processor power across devices.

Chapter 5

The Case for Software Adaptability

Now that we have established that there is significant variability across smartphone processors, we expect that we should be able to provide a proof of concept of an application that can adapt to this variability for the benefit of the device user and/or a service provider. One such application popularly applied in literature is *video playback* as mentioned in Section 2.5.2 since it is known to have many tunable parameters. The workflow for our experiments remains the same as in Section 3.3.

5.1 Application

We play a video encoded in three different formats (3gp, H.264 and Webm/VP8[1]) on each device [2]. In order to provide more choice in quality and power consumption, we also play the same videos at two or three different frame resolutions: 4CIF (704x576), CIF(352x288) and QCIF(176x144)[3]. For the purposes of our analysis we use one video across encodings and frame resolutions. Each frame resolution had its own raw video which we used to measure the quality of the videos of various encodings. The frame rate of encoding depends on the video selected and we keep the bit-rate constant in order to

[1] In this thesis, we will use Webm or VP8 interchangeably.

[2] These are among the only supported Android media formats. [21]

[3] These particular resolutions are chosen for two reasons. First, the 3gp wrapper only supports these resolutions and multiples thereof. Second, most well-known raw video sequences are only of these resolutions or VGA, which is not 3gp supported

maintain a constant file size across encodings. File sizes are kept constant for a particular video at a particular frame resolution. [4] The accepted standard for measuring the quality of a video is PSNR [32] [40]. We do this using the BVQM software [25] that compares the YUV format raw files obtained from each encoding to the original to generate a PSNR value. A higher PSNR is a better quality. More details about the process of encoding the videos from the raw files and obtaining a PSNR value are explained in the Appendix Section B.3.

For the purposes of our research, we used three different videos so that any observations we made and scenarios of adaptation we discussed could be applied across multiple videos or our hypothesis proved incorrect, and some videos did not provide any scope for adaptation or power savings. The details and specifications of the three videos are given in Table 5.1. The first video is a 19s long video containing a set of people ice-skating and is one of the well-known video sequences popularly used in video-related research. The second and third videos were obtained from the ITS supported Consumer Digital Video Library (CDVL) and can only be used for research and development. Each video has been edited to meet the ITS Video Quality Experts Group (VQEG) Multimedia Test Plan.

We present graphical results for two sets of runs: the first where we play *Ice* at 4CIF resolution, for which the total processor power across devices is shown in Figure 5.1 and second, where we play *Ice* at CIF resolution, for which the total processor power across devices is shown in Figure 5.2. The large difference in power between the CIF and 4CIF videos is because the 4CIF video has a significantly larger size and frame resolution. While the general ordering of the devices for any encoding based on power consumption does not change, it is still possible to alter the choice of which encoding to use for video

[4] This helps us assert that choosing a different video encoding will bear no additional network cost to a Service Provider or user.

Table 5.1. This table describes the details of the three different videos we will discuss in Chapter 5.

Ice [14]	
Description	Group of people ice-skating at a rink, with two in the foreground. Among the *well known* sequences used for video-related research.
Available Formats	4CIF (704x576, progressive) CIF (352x288, progressive) QCIF (176x144, progresive)
Chroma Sampling	4:2:2
Coding Complexity	Moderate
Origin and Quality	Unknown
Run Time	0:19
Copyright Restriction	None
NightStreet [24]	
Description	Night shot driving down street with camera held at an angle, showing bright lights on buildings and trees. Edited according to VQEG Multimedia Test Plan.
Available Formats	CIF (352x288, progressive) QCIF (176x144, progresive)
Chroma Sampling	4:2:2
Coding Complexity	Moderate
Origin and Quality	Original-Excellent
Run Time	0:12
Copyright Restriction	Research and Development
TeaHouse [24]	
Description	Interior of the Dushanbe Tea House, no scene cuts. Edited according to VQEG Multimedia Test Plan.
Available Formats	CIF (352x288, progressive) QCIF (176x144, progresive)
Chroma Sampling	4:2:2
Coding Complexity	Moderate
Origin and Quality	Original-Good
Run Time	0:12
Copyright Restriction	Research and Development

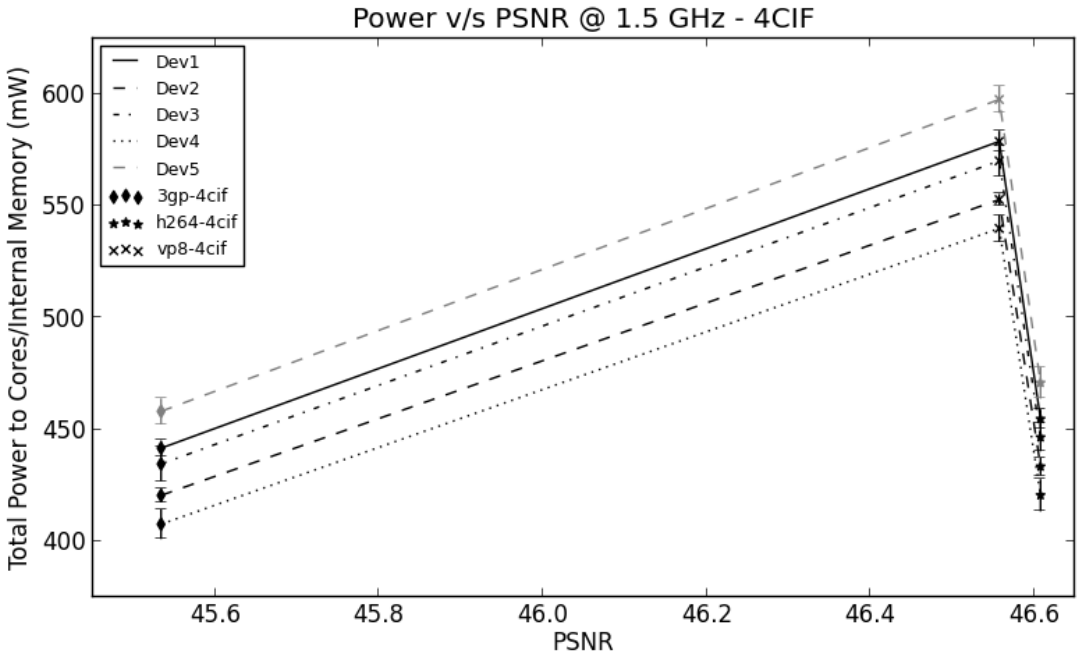

Figure 5.1. The figure shows Power versus PSNR for the playback of the *Ice* video with Frame Resolution 704x576. We observe that there is significant variability across devices at a particular encoding, denoted by a PSNR value for that encoding.

playback based on a specific power upper bound, as will be explained in the following section, using a comparison of the results we just presented for 4CIF and CIF resolutions.

A summary of the processor power variability across devices for the different frame resolutions is shown in Table 5.2. The purpose of the table is just to illustrate that the observations from Chapter 4 also hold for our new application scenario. A point to note here is that while the variability changes based on the frame resolution, the difference between processor power used for two different encodings on one device remains almost the same. This implies that any variability we do observe is not application-encoding induced, but device-induced.

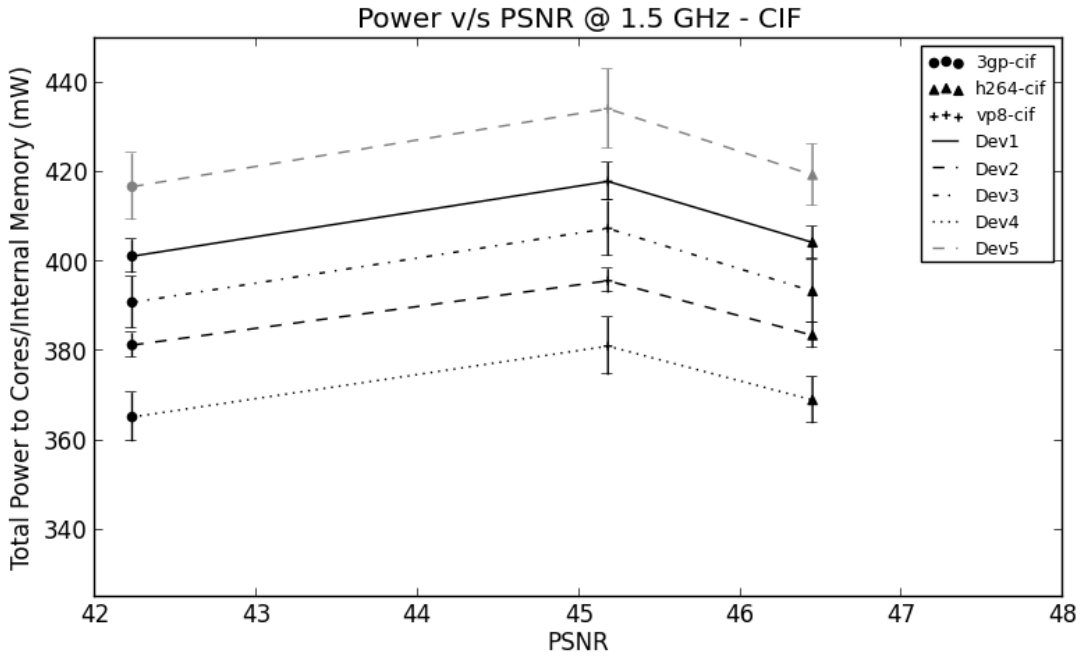

Figure 5.2. The figure shows Power versus PSNR for the playback of the *Ice* video with Frame Resolution 352x288. We observe that there is significant variability across devices at a particular encoding, denoted by a PSNR value for that encoding.

5.2 Potential for Adaptation

An important goal of the NSF Variability Expedition [1] has been to be able to leverage hardware variability by exposing it to the software to facilitate decision making for applications. This is the key principle expressed in the vision for UnO machines [16]. Therefore, in this section we will discuss some scenarios where it is possible for a service provider facilitating video playback on a device to adapt encodings and frame resolutions based on the knowledge of processor power variability across the devices for the benefit of the user.

If a service provider were to approach variability as a decision making parameter, we would need to consider two user requirements:

An upper bound for power consumption: Either the user or the service provider

Table 5.2. This table summarizes the processor power variability observed when we play the *Ice* video on the devices at three different frame resolutions, where each video is played in three different encoding formats. The devices run at 1.5GHz.

Encoding	Maximum Variability at QCIF	Maximum Variability at CIF	Maximum Variability at 4CIF
3gp	13.31%	14.10%	12.38%
H.264	13.64%	13.63%	11.95%
Webm	13.58%	17.64%	10.68%

should be able to specify an upper bound for the allowed energy consumption on the phone. Typically, a user would specify this in the only parameter of utmost importance to him/her, how long he/she would like the desired battery lifetime of the device to be. A service provider may want to restrict the power drawn on a user's phone so that the user gets desirable quality of service and they retain customers. Such a power goal is met when the phone draws power less than a specified value.

An acceptable quality lower bound: Different users have different assumptions about quality of service expected from certain applications. Thus, a user must be allowed to specify what their acceptable quality limits are. For video playback, any video that plays at a better quality than this will be acceptable to the user. For now, we measure quality by PSNR as expressed above. However, besides the objective PSNR measurement, we could also use higher frame resolution to be a more desired quality for video playback.

In an argument very similar to ours, Martins et. al. [29] create an abstraction called *Application Modes* where power savings are achieved through graceful degradation. Developers create different modes for an application by selecting sets of functionalities that entail different application behaviors, as perceived by the user, in exchange for reduced energy consumption. Graceful degradation is achieved through various ways: different settings, different algorithms, even different programs. We use the same argu-

ment that the user must decide a preference for a trade-off between the two goals listed above, since it may not always be possible to meet both.

We must thus target those encodings that reside in that sweet spot at the intersection of the area of a power versus PSNR graph under the quality and power bounds. This allows us to reason about the trade-offs between quality and energy consumption. Figure 5.3 and Figure 5.4 shows the power versus PSNR values for both the high and medium resolution video, which differ only in size and quality. We will now use these figures to illustrate how we would meet the two requirements once obtained from the user.

5.2.1 Sample Scenarios

In this section, we will discuss some scenarios where we could leverage device variability for energy efficiency.

Scenario 1: Figure 5.3 plots the power versus PSNR values for devices playing the *Ice* video in three different encoding formats at two different frame resolutions. Clearly, the 4CIF frame resolution is qualitatively and quantitatively better than the CIF frame resolution videos. However, the graph does show that the quality and power values overlap, thus providing the service provider with a wider range of choices to offer to the user. Consider, for example, that the user sets a power goal of 430mW as shown by the gray dotted line in the figure. If a service provider could equip themselves with a video tuned to these different encodings and frame resolutions, for Dev4, we would be able to play the best available quality, with the highest frame resolution – here, the *Ice* video encoded in the H.264 format at a 4CIF resolution.

However, all the other devices would only be able to play the video encoded using the H.264 encoding format at a much lower frame resolution – here, CIF. While this would result in subjective viewing differences, it would considerably save power on Dev1-3 and Dev5 (between 12-13% power savings) while also meeting the power goal

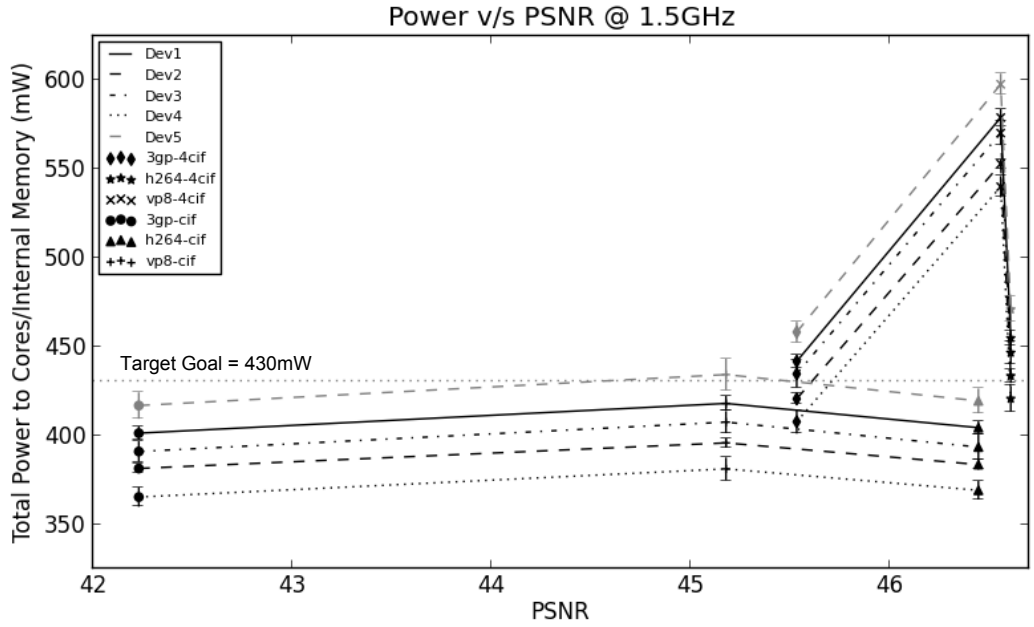

Figure 5.3. When we analyze the two Power versus PSNR plots above together, we can make a choice between frame resolution and encoding for a preferred quality and power goal. In this scenario, with a power goal of 430mW, we find that we can offer the best quality within that goal to Dev4 with an H.264 encoding video at a high frame resolution, while the video used for other devices differs in encoding, quality and frame resolution.

set by the user which would have disallowed Dev1-3 and Dev5 from playing any videos that exceeded the user-set power goal of 430mW. This is in line with the idea of *graceful degradation* where the application decides to compromise on the quality of service in order to meet a particular requirement specified by the user.

Scenario 2: Now, let us assume, for the purposes of this analysis, that service providers currently use a static setting for all users. The popular encoding to use is VP8/Webm since it is supposed to provide optimal compression. Let us assume then that service providers statically select the Webm encoding format at a CIF frame resolution, irrespective of what kind of device the user may use to connect to the service. Assume the user now has a power goal of 400mW. Figure 5.4 indicates the power goal as a gray

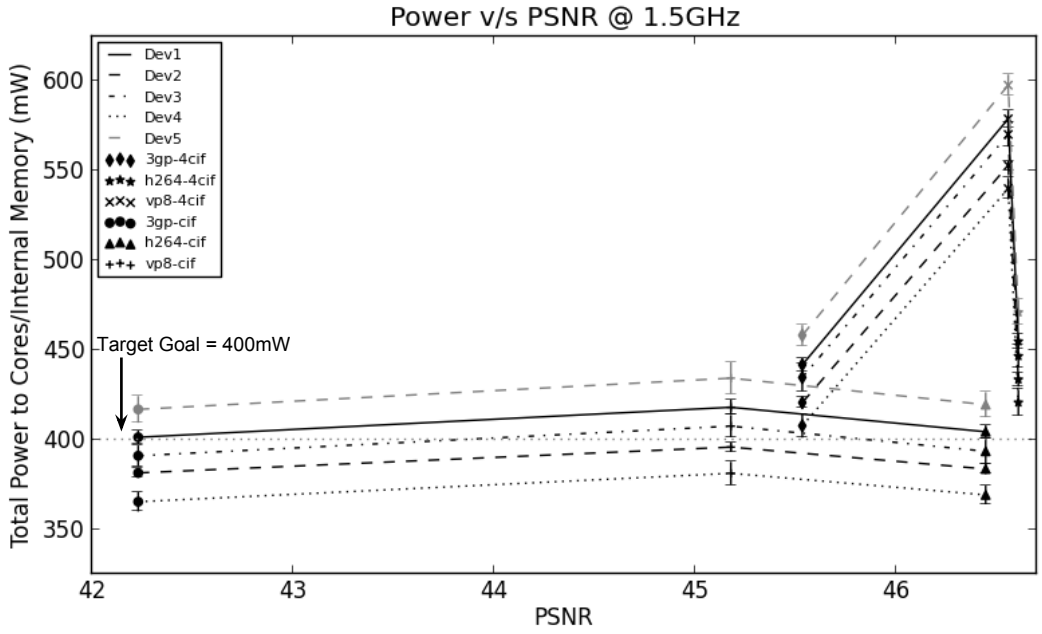

Figure 5.4. When we analyze the two Power versus PSNR plots above together, we can make a choice between frame resolution and encoding for a preferred quality and power goal. In this scenario, with a power goal of 400mW, we find that Dev2-4 can benefit from a higher quality video encoding format while the other devices would have to compromise on quality and frame resolution.

dotted line while showing the power values for playing *Ice* in three different encoding formats at two different frame resolutions.

With a power goal of 400mW, it is possible to encode Dev2-4 using the H.264 encoding format and play a better quality video than the original static choice of Webm/VP8 and thus save around 3% in power consumed[5], which is a win-win situation for the user and provider. Dev1 and Dev5, on the other hand, in a non-static decision-making environment would have to compromise on their quality to a large extent, and play a video of poorer encoding format and reduced frame resolution further to reduce power consumption to meet the goal, which is still graceful degradation, as opposed to returning an error

[5] 3% of power savings is equivalent, on an average, to about 15 minutes of battery lifetime, which is significant for the user, especially when the application in question only plays a 19s video.

Table 5.3. Difference between playing CIF and QCIF frame resolutions for different encodings, expressed as a percentage of the lower of values of power consumed at each device for a particular encoding. In scenario 3, we discuss how these values can be used by service providers to determine what encoding format and frame resolution to play a video in for a particular device.

Encoding Format	Dev1 (%)	Dev2 (%)	Dev3 (%)	Dev4 (%)	Dev5 (%)
3gp	1.12	0.36	1.17	0.001	3.36
H.264	0.95	0.30	0.91	0.018	2.18
vp8	2.85	4.44	4.44	5.36	3.58

to the user saying the video they requested could not be played within their power goals. One other example of embracing diversity shown in Figure 5.4 is the fact that Dev3 can play Webm at a lower frame resolution for almost the same power that Dev4 can play 3gp at a much higher frame resolution. If this were known to a service provider, it may be the case that they decide to play the video encoded in 3gp at 4CIF on Dev4 and in Webm at CIF on Dev3 while both devices would then have a similar battery lifetime.

Scenario 3: For this scenario, we consider the *NightStreet* video played on all the devices running at 1.5GHz, at two different frame resolutions, CIF and QCIF, and in three different encoding formats encoded at a constant frame and bit rate. Table 5.3 shows the differences between the power used by the phone processors during QCIF and CIF playback. Based on the values in the table, we can make a general rule such that if the difference between playing CIF over QCIF exceeds 3%, we will recommend to the service provider to supply the user with a QCIF format of the video, otherwise, it is safe to play the CIF video. Thus, for some devices and encodings (Dev 1 to 4, 3gp and H.264), we suggest to the user that playing the CIF frame resolution would result in better subjective video quality with a higher frame resolution and not that great of a power loss as compared to playing the same video at the same encoding with the QCIF frame resolution. Thus, we can provide an improved quality for a reasonable increase in power

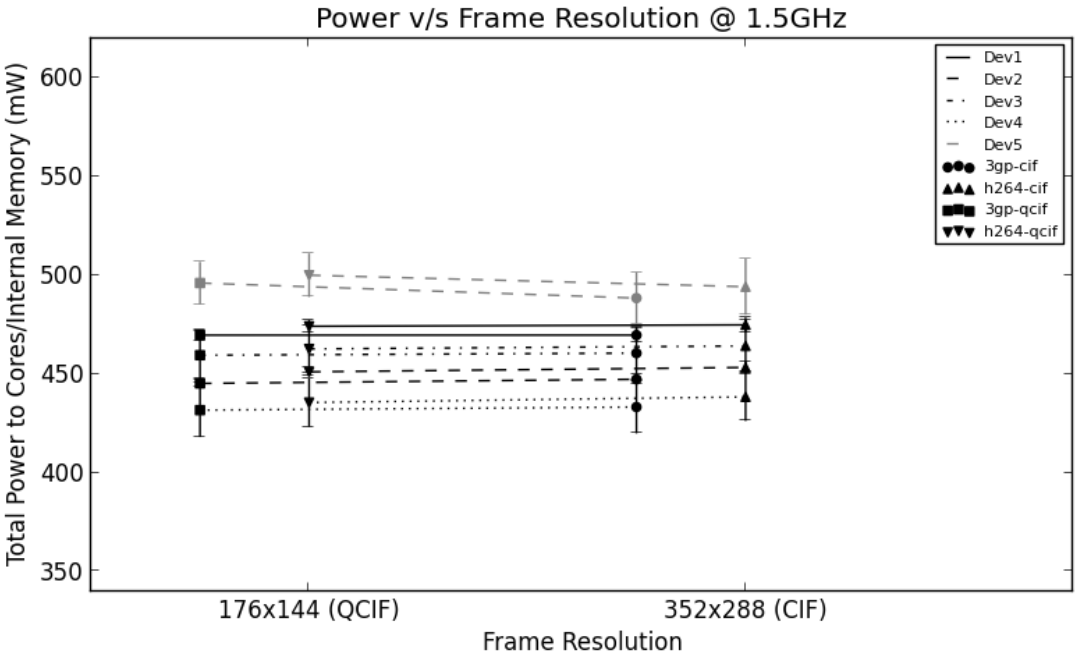

Figure 5.5. This figure shows the total processor power for the devices when they play the *Tea House* video in three different encoding formats, 3gp, H.264 and Webm and two different frame resolutions, QCIF(176x144) and CIF(352x288). The devices operate at 1.5GHz. There is barely any difference in power across the frame resolution.

draw. On the other hand, for Dev5, it seems like a bad idea to be playing the 3gp encoding at a CIF resolution since the power wasted by playing CIF would not be worth the cost to the phone of playing at that higher resolution. Similarly, it would be recommended to play a CIF frame resolution for the Webm video on Dev1 since it is below the 3% cutoff whereas for all the other devices, it would be recommended to play a QCIF frame resolution for the Webm video. Similar such cut-offs can be algorithmically determined for the *Ice* video, that could result in upto 30% power savings when playing the video at a drastically lower frame resolution[6]. One downside with this kind of decision-making however is that service providers or device manufacturers would need to provide some kind of representative data on the different kinds of devices one could encounter.

[6]If we performed a similar analysis on the data for playing *Ice* at CIF and 4CIF frame resolutions, we find around 30% savings by playing a lower frame resolution.

Scenario 4: This idea of leveraging variability based on the differences across frame resolutions however, does not apply uniformly to all types of videos. Consider, for example, the *TeaHouse* video. It is a rather static video of a tea room, where a lot of the consecutive frames are very similar to each other. There is barely any difference in the processor power across the devices for the two different frame resolutions and making a dynamic choice is unlikely to provide any energy savings. This is an example of a situation where we are unable to reap any benefits from the processor power variability across the devices.

Chapter 6

Implications, Future Work and Conclusion

Having characterized processor power variability in Chapter 4 and discussed in detail some sample scenarios where we can try to leverage power variability for energy efficiency, in this chapter, we sum up our findings by explaining what we believe are some of the implications of our work, how this work could be expanded upon in the future and finally, conclude by summarizing our results and observations.

6.1 Implications

6.1.1 Instrumentation

As Martins et. al. [29] aptly state, energy profiling and forecasting is one of the key challenges of adapting applications on mobile devices for energy efficiency. An important aspect of such profiling is instrumentation – to obtain appropriate information from the devices at the right granularity in real-time in order to make dynamic adaptive decisions. While this thesis as well as recent work by Balaji et. al. [2] have extensively used equipment provided by hardware manufacturers for measurement and debugging, it is important to be able to classify devices on a power scale based on sensor readings. For example, it was clear from our experiments that Dev4 was a low power device, and

with appropriate instrumentation, we may be able to provide the user of Dev4 much better quality at no cost to his device, while maintaining some average battery lifetime goal. An avenue of study would be to instrument the phones with some kind of energy sensor that would allow application service providers to bin a phone based on its average power draw. This function of procuring power measurements from the phone is currently performed by our Debug Board, and if the hardware manufacturers could instrument the phone with a way to measure these values without building sophisticated external hardware attachments, it would certainly incentivize applications to embrace the diversity in devices.

6.1.2 Variability-Aware Adaptation

By factoring in user preferences for an acceptable lower quality limit and an acceptable battery drain or power usage upper limit, it is possible to make different video playback decisions in terms of encoding format, frame resolution, and bit rate, for some videos such that these decisions differ across devices. This is thus an example of the case where applications can embrace the diversity of devices and still provide quality of service to the user.

6.1.3 Service Provider Adoption

While advocating for user-interactive control of their battery lifetime, it becomes important to specify what arrangements service providers may need in order to write, develop and use such applications. In the scenarios we discussed in Section 5.2.1, it was possible for service providers to make decisions to save phone energy or improve quality with a small set of encoding formats and frame resolutions. It will be challenging however to assess how well such tunable parameters scale. However, since many video-based service providers already provide on-the-fly real-time bit-rate changes, it may not be that

hard to spread adoption of variability-aware video playback.

6.1.4 Potential User Interactions

Given the fact that we are dealing with variability in *smartphones*, it would be important to predict some cases where this kind of variability aware energy efficiency would directly impact a real-world user of the technology. Here are some examples. When a user buys a new phone, the manufacturer may recommend he/she visit their website (since they already do that) to enable some settings that would allow them to identify the user's phone and facilitate energy savings on the user's phone when he/she is running some specific types of applications with a low impact on his/her perceived quality of the application on his/her phone.

Based on the differences in variability we observed caused by ambient temperature changes in Section 4.2, we can envision another scenario. If a user was watching a video outside on a warm day, they may experience a particular quality of service. Now, if they were to watch the same video in an indoor, climate-controlled environment at much cooler temperatures, they may perceive much better quality of service with minimal battery drain, i.e., no compromise on any power goals the user may have set.

6.2 Future Work

6.2.1 More Adaptive Applications

While the foundation for our results on observed variability was based on running standard benchmarks, and the adaptation explained using a proof-of-concept application, it will be interesting to see if there are other applications that could also leverage processor power variability in mobile devices for energy efficiency. Flinn et. al. [12] demonstrated their energy efficiency ideas using speech recognition and navigation applications as well as the web browser application. Martins et. al. [29] also demonstrate subjective

differences in application behavior using navigation and audio-visual playback. Another application that can make dynamic adaptive algorithmic decisions for energy efficiency is file compression [3]. Thus, some of these applications should be tested to leverage processor power variability in mobile devices.

6.2.2 Analyzing Graceful Degradation

With the scenarios discussed in Section 5.2.1, we only presented a proof-of-concept for energy savings by dynamic decision making. In order to handle degraded quality of service gracefully, we must devise an algorithm for deciding when to drop the user's quality preference over their power preference. One way to do this may be to try and assign a composite power-quality metric to each device and application such that the decision over the trade-off between quality and power is determined by the priority preferences of the user. We must also provide some way for the user or the service provider to specify a power goal or specify a general power difference cut-off as we discussed in Scenario 2 in Section 5.2.

6.3 Conclusion

As semiconductor manufacturers build smaller components, circuits and chips at that scale become less reliable and more expensive to produce, and no longer conform to the rigid hardware specifications usually expected of them. While the scope of this behavior has previously been limited to hardware manufacturers, a push has now been made to embrace device variability rather than hide it by increasing gaurdbands applied to chip designs. Our work uses a software-systems approach to analyze and characterize processor power variability in mobile devices, since mobile devices have replaced personal computers and laptops as the next generation of general purpose computing devices.

Using smartphones as representative mobile devices, we have explored the nature of processor power variability in these devices, and in the process presented results for the significance of such variability. In our analysis and characterization of processor power variability, we observed processor power variability across smartphones in the range 6%-15% at different frequencies of operation, which could mean upto 70 minutes of improved battery lifetime. We characterize this variability as a factor of contributions from the processor core, the digital core and the internal memory and discuss the significance of idle power variability. We also test that processor power variability across mobile devices is amplified with increasing temperatures. We observe that the processor power variability across devices that were all operating at 1.5GHz, is amplified from a range of 6%-10% at normal temperatures (27°C to 33°C) to 14%-17% at warmer temperatures (50°C to 55°C) for the CPU-intensive benchmarks we used.

We also present a proof-of-concept application to leverage the observed patterns of variability as a case of adapting software to deal with variability in smartphones or mobile devices. By setting a power goal and/or a quality goal, it is possible for us to proactively choose the encoding, and some other tunable parameters to use for video playback, resulting in potential energy savings of 3%-13% or an improved battery lifetime of an hour over the average expected 8 hours.

Appendix A

Additional Figures and Observations

A.1 Power Measurements For 8 Channels

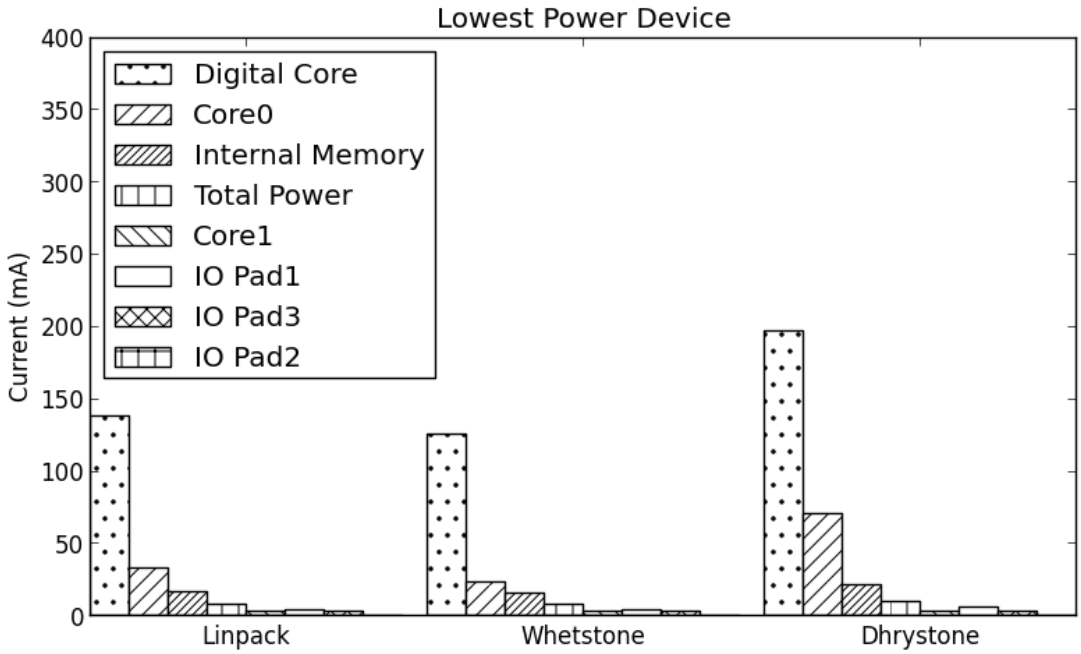

Figure A.1. Power measurements for 8 channels for the device that consumes the least power for most channels. For most channels, the values are too low to make any contribution to the total variability in processor power.

We observe from Figures A.1 and A.2 that the power consumed by channels like Core1 and IO Pad 1-3 are too low to contribute to the overall processor power variability

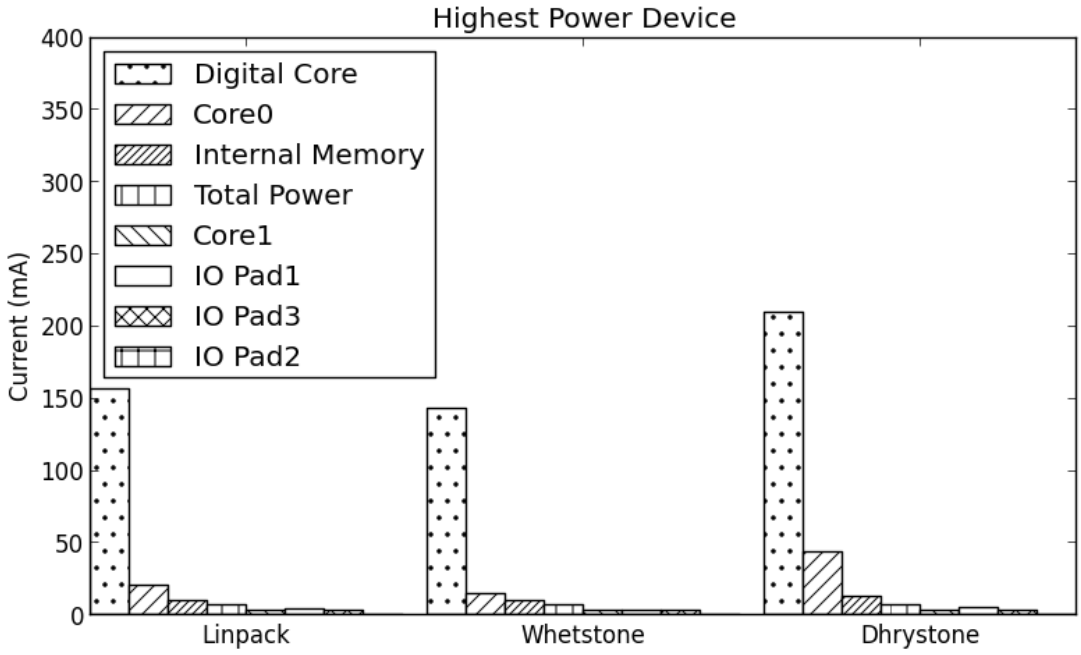

Figure A.2. Power measurements for 8 channels for the device that consumes the most power for most channels. For most channels, the values are too low to make any contribution to the total variability in processor power.

and are relatively the same across devices. The two figures show the power consumption measured for 8 rails in terms of the current for the device that consumes the least and most power for most of the channels. The *Total Power* channel is a rail provided by the WQEPM software. However, it does not actually give us the sum of the power to various channels, and may be irrelevant because the Debug Board acts as a battery emulator and the *Total Power* rail draws its measurements from the battery power. For *IO Pad3*, the values are of the order 0.3mA and thus are not visible in the figures.

A.2 Run to Run Variation

Figure A.3 shows that for most applications, the power consumption on the device increases over time, most likely because the phone temperature increases slightly over time when the phone is repeatedly used.

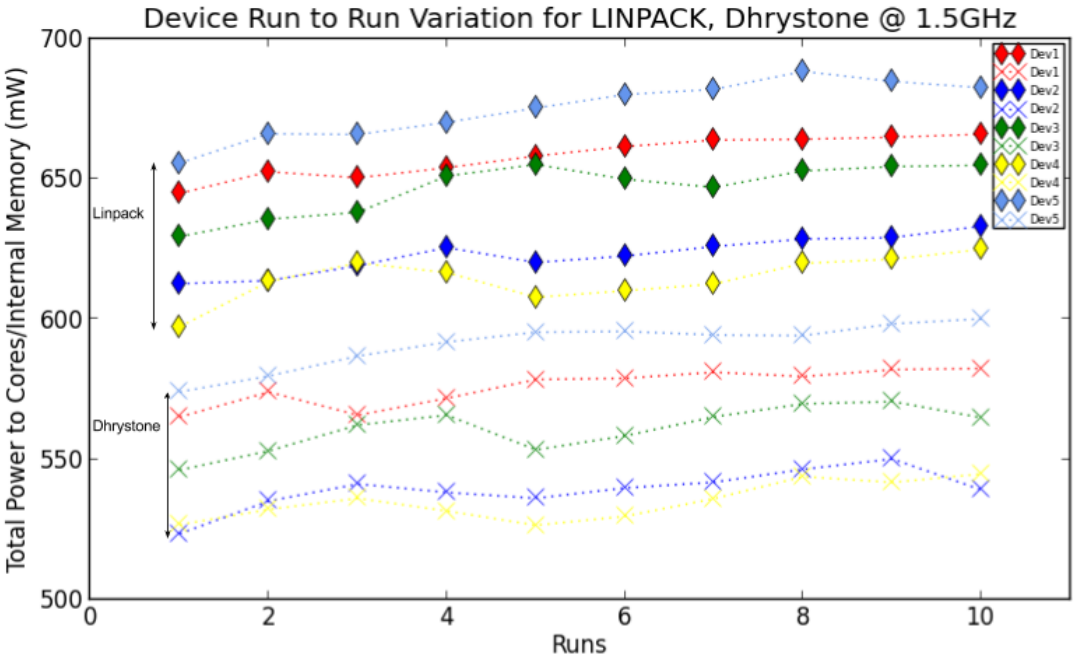

Figure A.3. Run to run variation across 10 runs on the devices running LINPACK and Dhrystone.

A.3 Encodings and Processor Frequencies

In some of our initial experiments, it was observed that vp8 would aggressively draw power at high frequencies but would be much more conservative at lower frequencies. There appears to be some kind of point of inflexion after which vp8 can play videos smoothly at higher power as opposed to the choppy blocks it displays at lower frequencies. This inflexion is shown in the second graph in Figure A.4. The first graph shows us the reason for this is the much larger increase in core0 current supply to vp8 videos than to other encodings. While irrelevant to our discussion on device variability, it seemed like an interesting observation.

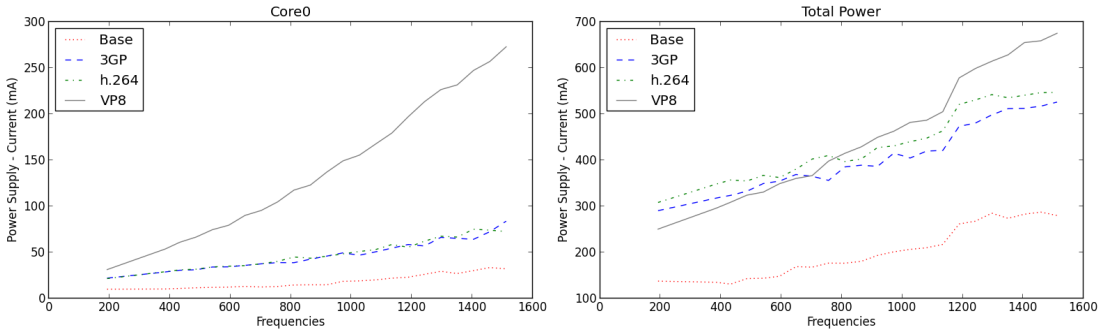

Figure A.4. The current supplied is plotted against the frequency of the processor to show the trend for each encoding as frequency changes.

A.4 Calculation of Margin of Error

Let P be the total power and the standard deviation, σ be $s\%$ of the total power. We know that the number of runs for our experiment, n is 10.

For a confidence interval of 95%, $z^* = 1.96$, and the margin of error is

$$m = \frac{z^*\sigma}{\sqrt{n}}$$

$$m = \frac{1.96 * s * P}{\sqrt{10}}$$

For our CPU-intensive benchmarks at normal ambient temperatures, we observed a maximum standard deviation of about 2% of the total power, and if $s = 0.02$, the margin of error, $m = 0.012P$, which is 1.2% of the total power. Similarly, at warmer ambient temperatures, we observe a maximum standard deviation of 4% of the total power. Here, $s = 0.04$, and $m = 0.024P$. The margin of error is 2.4% of the total power.

Consider the maximum variability we observed in Table 4.2 for Internal Memory. The maximum standard deviation was 6% of the power measured for the Internal Memory channel. The margin of error here is 3.7% of that power.

Appendix B

Other Workflow Details

B.1 NSF Demo Documentation

Last fall, we prepared a demonstration for the NSF Variability Expedition review. This section lists the packages needed and the workflow to recreate such a demonstration in the future. The setup was meant to show by very short runs across two devices that there was observable variability across the two devices.

B.1.1 Software Packages

The demo will only work on a Windows machine since the debug board is not supported on non-Windows systems. The following software packages are required on any system that wishes to setup the demo: the NRT MiniBoard driver, VisualC++ for WQEPM, the Android USB Driver with modifications to support the MSM8660 phones, SQLite3, ActiveState Perl with CSV support and Boomslang (matplotlib, and associated dependencies) for supporting a graphical view.

B.1.2 Workflow

The workflow has already been scripted and the script will be made available on request.

1. Start WQEPM and start the browser with the default URL associated with WQEPM.
2. Run the script to start monitoring the channels specified. Channel IDs are obtained from the WQEPM configuration file.
3. Run the experiments you would like to test. Keep them short, since the monitoring script times out.
4. Stop WQEPM to be release the database lock.
5. Extract data for that test, obtain average current for the channels being monitored, plot the values and display the graph.
6. Repeat this for a second device connected to a second debug board, and display the two graphs together to observe the channel-wise variability across the two devices.

B.2 ADB Commands

This is a list of some of the ADB commands that we used to control the frequency and core usage on the devices.

Turning Core 1 Off:

```
adb stop mpdecision
adb shell echo 0 /sys/devices/system/cpu/cpu1/online
```

Setting Frequency:

```
adb shell echo userspace > \
 /sys/devices/system/cpu/cpu0/cpufreq/scaling_governor
adb shell echo 192000 > \
 /sys/devices/system/cpu/cpu0/cpufreq/scaling_min_freq
```

```
adb shell echo 192000 > \
 /sys/devices/system/cpu/cpu0/cpufreq/scaling_max_freq
```

B.3 Video Quality Measurements

As mentioned in Section 5.1, we measure video quality in terms of a PSNR value by first encoding the raw video in the formats we plan to use and comparing the decoded raw video output from these videos with the original raw video.

B.3.1 Using ffmpeg

We used ffmpeg to encode and decode the videos. A standard ffmpeg encoding command to ensure constant bit-rate, which in turns ensures constant file size looks like this:

```
ffmpeg -r 29.97 -i ntia_tea4-qcif_original.avi -r 29.97 \
 -s 176x144 -c:v libvpx -preset medium -b:v 500k -minrate 500k \
 -maxrate 500k -bufsize 500k -pass 1 -an -f webm /dev/null \
 && ffmpeg -r 29.97 -i ntia_tea4-qcif_original.avi \
 -s 176x144 -r 29.97 -c:v libvpx -preset medium -b:v 500k \
 -minrate 500k -maxrate 500k -bufsize 500k -pass 2 \
 -an ntia_tea4-qcif_webm.webm
```

We use a 2 pass encoding procedure, and set constant frame rate using the $-r$ option and the required video codec using $c : v$. Here, *libvpx* indicates that we are using vp8 or the webm encoding format. $-s$ indicates the frame resolution. [36] The frame rate depends on the video selected and we keep the bit-rate constant in order to maintain a constant file size across different encodings. The bit-rate is chosen based on receommendations from

YouTube support on appropriate bit rates for specific frame resolutions. [19] A standard decoding command to obtain the raw video looks like this:

```
ffmpeg -i ntia_tea4-qcif_webm.webm \
   -f rawvideo ntia_tea4-qcif_webm.yuv
```

B.3.2 BVQM

We use the BVQM software [25] to compare these raw videos to obtain a PSNR value. BVQM only accepts data of the format *name_scene_hrc.yuv*, where name and scene are constant values for our purposes. We name our files name_scene_3gp, etc and compare the videos to name_scene_original.yuv obtained from [14] or [24]. For more details on the use of the BVQM software, refer to the Batch Video Quality Metric User Manual [23].

Bibliography

[1] NSF Variability Expedition. http://variability.org.

[2] Bharathan Balaji, John McCullough, Rajesh K. Gupta, and Yuvraj Agarwal. Accurate characterization of the variability in power consumption in modern mobile processors. In *Proceedings of the 2012 USENIX conference on Power-Aware Computing and Systems*, HotPower'12, pages 8–8, Berkeley, CA, USA, 2012. USENIX Association.

[3] Kenneth Barr and Krste Asanović. Energy aware lossless data compression. In *Proceedings of the 1st international conference on Mobile systems, applications and services*, MobiSys '03, pages 231–244, New York, NY, USA, 2003. ACM.

[4] Luis Angel D. Bathen, Nikil D. Dutt, Alex Nicolau, and Puneet Gupta. Vamv: Variability-aware memory virtualization. In Wolfgang Rosenstiel and Lothar Thiele, editors, *DATE*, pages 284–287. IEEE, 2012.

[5] Luis Angel D. Bathen, Mark Gottscho, Nikil Dutt, Alex Nicolau, and Puneet Gupta. Vipzone: Os-level memory variability-driven physical address zoning for energy savings. In *Proceedings of the eighth IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis*, CODES+ISSS '12, pages 33–42, New York, NY, USA, 2012. ACM.

[6] K.A. Bowman, S.G. Duvall, and J.D. Meindl. Impact of die-to-die and within-die parameter fluctuations on the maximum clock frequency distribution for gigascale integration. *Solid-State Circuits, IEEE Journal of*, 37(2):183–190, 2002.

[7] Liang Cheng, S. Mohapatra, M.E. Zarki, N. Dutt, and N. Venkatasubramanian. A backlight optimization scheme for video playback on mobile devices. In *Consumer Communications and Networking Conference, 2006. CCNC 2006. 3rd IEEE*, volume 2, pages 833–837, 2006.

[8] Jeonghwan Choi, Chen-Yong Cher, Hubertus Franke, Henrdrik Hamann, Alan Weger, and Pradip Bose. Thermal-aware task scheduling at the system software level. In *Proceedings of the 2007 international symposium on Low power electronics and design*, ISLPED '07, pages 213–218, New York, NY, USA, 2007. ACM.

[9] H J Curnow, B A Wichmann, and Tij Si. A synthetic benchmark. *The Computer Journal*, 19:43–49, 1976.

[10] S. Dighe, S. Vangal, P. Aseron, S. Kumar, T. Jacob, K. Bowman, J. Howard, J. Tschanz, V. Erraguntla, N. Borkar, V. De, and S. Borkar. Within-die variation-aware dynamic-voltage-frequency scaling core mapping and thread hopping for an 80-core processor. In *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2010 IEEE International*, pages 174–175, 2010.

[11] Jack J. Dongarra, Piotr Luszczek, and Antoine Petitet. The linpack benchmark: Past, present, and future. concurrency and computation: Practice and experience. *Concurrency and Computation: Practice and Experience*, 15:2003, 2003.

[12] Jason Flinn and M. Satyanarayanan. Energy-aware adaptation for mobile applications. *SIGOPS Oper. Syst. Rev.*, 33(5):48–63, December 1999.

[13] Jason Flinn and M. Satyanarayanan. Powerscope: A tool for profiling the energy usage of mobile applications. In *Proceedings of the Second IEEE Workshop on Mobile Computer Systems and Applications*, WMCSA '99, pages 2–, Washington, DC, USA, 1999. IEEE Computer Society.

[14] Xiph.Org Foundation. Sample Videos Collection. http://media.xiph.org/video/derf/.

[15] M. Gottscho, A.A. Kagalwalla, and P. Gupta. Power variability in contemporary drams. *Embedded Systems Letters, IEEE*, 4(2):37–40, 2012.

[16] P. Gupta, Y. Agarwal, L. Dolecek, N. Dutt, R.K. Gupta, R. Kumar, S Mitra, A. Nicolau, T.S. Rosing, M.B. Srivastava, S. Swanson, and D Sylvester. Underdesigned and opportunistic computing in presence of hardware variability. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 32(1):8–23, 2013.

[17] Juan Rubio Soraya Ghiasi Heather Hanson, Karthick Rajamani and Freeman Rawson. Benchmarking for power and performance.

[18] Apple Inc. iPhone Lithium-Polymer Batteries. http://www.apple.com/batteries/iphone.html.

[19] Google Inc. Youtube Support - Advanced Encoding Settings. http://support.google.com/youtube/bin/answer.py?hl=en&answer=1722171.

[20] Google Inc. and the Open Handset Alliance. Android Debug Bridge. http://developer.android.com/tools/help/adb.html.

[21] Google Inc. and the Open Handset Alliance. Android Supported Media Formats. http://developer.android.com/guide/appendix/media-formats.html.

[22] ITRS. The International Technology Roadmap for Semiconductors. http://public.itrs.net/.

[23] ITS. Batch Video Quality Metric (BVQM) User's Manual. http://www.its.bldrdoc.gov/publications/2558.aspx.

[24] ITS. Consumer Digital Video Library. http://www.cdvl.org/.

[25] ITS. Institute for Telecommunication Sciences Technical Progress Report. http://www.its.bldrdoc.gov/publications/2676.aspx.

[26] R.W. Johnson, J.L. Evans, P. Jacobsen, J.R. Thompson, and M. Christopher. The changing automotive environment: high-temperature electronics. *Electronics Packaging Manufacturing, IEEE Transactions on*, 27(3):164–176, 2004.

[27] K. Lee, N. Dutt, and N. Venkatasubramanian. An experimental study on energy consumption of video encryption for mobile handheld devices. In *Multimedia and Expo, 2005. ICME 2005. IEEE International Conference on*, pages 1424–1427, 2005.

[28] Roy Longbottom. Android Benchmark Apps. http://www.roylongbottom.org.uk/android%20benchmarks.htm.

[29] Marcelo Martins and Rodrigo Fonseca. Application modes: a narrow interface for end-user power management in mobile devices. In *Proceedings of the 14th Workshop on Mobile Computing Systems and Applications*, HotMobile '13, pages 5:1–5:6, New York, NY, USA, 2013. ACM.

[30] Dustin McIntire, Thanos Stathopoulos, Sasank Reddy, Thomas Schmidt, and William J. Kaiser. Energy-efficient sensing with the low power, energy aware processing (leap) architecture. *ACM Trans. Embed. Comput. Syst.*, 11(2):27:1–27:36, July 2012.

[31] Shivajit Mohapatra, R. Cornea, Hyunok Oh, K. Lee, Minyoung Kim, N. Dutt, R. Gupta, A. Nicolau, S. Shukla, and N. Venkatasubramanian. A cross-layer approach for power-performance optimization in distributed mobile systems. In *Parallel and Distributed Processing Symposium, 2005. Proceedings. 19th IEEE International*, pages 8 pp.–, 2005.

[32] A.K. Moorthy, Lark Kwon Choi, A.C. Bovik, and G. De Veciana. Video quality assessment on mobile devices: Subjective, behavioral and objective studies. *Selected Topics in Signal Processing, IEEE Journal of*, 6(6):652–671, 2012.

[33] Earl Oliver and Prof S. Keshav. Data driven smartphone energy level prediction.

[34] Jeongyeup Paek, Joongheon Kim, and Ramesh Govindan. Energy-efficient rate-adaptive gps-based positioning for smartphones. In *Proceedings of the 8th international conference on Mobile systems, applications, and services*, MobiSys '10, pages 299–314, New York, NY, USA, 2010. ACM.

[35] A. Pant, P. Gupta, and M. van der Schaar. Appadapt: Opportunistic application adaptation in presence of hardware variation. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 20(11):1986–1996, 2012.

[36] FFmpeg Project. FFMPEG Documentation. http://ffmpeg.org/ffmpeg.html.

[37] Qualcomm. Snapdragon Mobile Development Platform Legacy Devices. https://developer.qualcomm.com/mobile-development/development-devices/snapdragon-mdp-legacy-devices.

[38] Qualcomm. WQEPM User Guide. Covered under NDA.

[39] Chris Taylor. Smartphone Sales Overtake PCs for the First Time. http://mashable.com/2012/02/03/smartphone-sales-overtake-pcs/.

[40] Y. Wang. Survey of objective video quality measurements, 2006.

[41] Yunfeng Wang, Qiang Wu, and Wei Xie. Hardware-software co-design for dynamic reconfigurable computing with collaborative supports of architecture and operating system. In *Computer Supported Cooperative Work in Design, 2007. CSCWD 2007. 11th International Conference on*, pages 275–279, 2007.

[42] L. Wanner, C. Apte, R. Balani, P. Gupta, and M. Srivastava. Hardware variability-aware duty cycling for embedded sensors. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 21(6):1000–1012, 2013.

[43] L. Wanner, R. Balani, S. Zahedi, C. Apte, P. Gupta, and M. Srivastava. Variability-aware duty cycle scheduling in long running embedded sensing systems. In *Design, Automation Test in Europe Conference Exhibition (DATE), 2011*, pages 1–6, 2011.

[44] Reinhold P. Weicker. Dhrystone: a synthetic systems programming benchmark. *Commun. ACM*, 27(10):1013–1030, October 1984.

[45] Rui Zheng, J. Velamala, V. Reddy, V. Balakrishnan, E Mintarno, S Mitra, Srikanth Krishnan, and Yu Cao. Circuit aging prediction for low-power operation. In *Custom Integrated Circuits Conference, 2009. CICC '09. IEEE*, pages 427–430, 2009.

[46] Jun Zhou, D. Kinniment, G. Russell, and A. Yakovlev. Adapting synchronizers to the effects of on chip variability. In *Asynchronous Circuits and Systems, 2008. ASYNC '08. 14th IEEE International Symposium on*, pages 39–47, 2008.