Workload Characterization on Android Based Intel Devices

Major Project Report

Submitted in partial fulfillment of the requirements

for the degree of

Master of Technology

in

Electronics & Communication Engineering

(Communication Engineering)

By

Komal Shah

(12MECC25)

Electronics & Communication Engineering Branch
Electrical Engineering Department
Institute of Technology
Nirma University
Ahmedabad - 382481
May, 2014

Workload Characterization on Android Based Intel Devices

Major Project Report

Submitted in partial fulfillment of the requirements

for the degree of

Master of Technology

in

Electronics & Communication Engineering

(Communication Engineering)

By

Komal Shah

(12MECC25)

Under the Guidance of

Mr. Ajaya Durg
Principle Engineer, Intel

Dr. Dhaval Pujara
Professor, EC, IT, NU

Electronics & Communication Engineering Branch
Electrical Engineering Department
Institute of Technology
Nirma University
Ahmedabad - 382481
May, 2014

Declaration

This is to certify that

a. This thesis comprises my original work towards the degree of Master of Technology in Communication Engineering at Nirma University and has not been submitted elsewhere for a degree.

b. Due acknowledgement has been made in the text to all other material used.

- Komal Shah

(12MECC25)

Certificate

This is to certify that the project entitled **"Workload Characterization on Android Based Intel Devices"** submitted by **Komal Shah (12MECC25)**, towards the partial fulfillment of the requirements for the degree of Master of Technology in Communicaton Engineering of Nirma University, Ahmedabad is the record of work carried out by her under our supervision and guidance. In our opinion, the submitted work has reached a level required for being accepted for examination. The results embodied in this major project, to the best of our knowledge, haven't been submitted to any other university or institution for award of any degree or diploma.

Date: Place: Ahmedabad

Guide

Dr. Dhaval Pujara
(Professor, EC)

Program Coordinator

Dr. D. K. Kothari
(Professor, EC)

HOD, EE

Dr. P. N. Tekwani
(Professor, EE)

Director

Dr. K. Kotecha
(Director, IT, NU)

Acknowledgements

I offer my sincere thanks to the very individual who played their possible role in inspiring me, motivating me and helping me during the project.

I am deeply indebted to my guide Dr. Dhaval Pujara and industrial guide Mr. Ajaya Durg, Principal Engineer, Intel Corporation for their constant guidance, motivation and support. They have devoted significant amount of valuable time to plan and discuss the thesis work. Without their experience and insights, it would have been very difficult to do quality work.

I would like to express my gratitude and sincere thanks to Dr. P. N. Tekwani, Head of Electrical Engineering Department and Dr. D. K. Kothari, Coordinator, M. Tech Communication Engineering program for allowing me to undertake this internship.

I also wish to thank Paul S. Chelladurai, Manoj Dawarwadikar, Shareef Hakim, Arojit Roychowdhury and Sundar Iyer for their help, support and guidance. Without their guidance and support, it would have been very difficult to complete the project.

I wish to thank my friends for their delightful company which kept me in good humor throughout the year.

Last, but not the least, no words are enough to acknowledge constant support and sacrifices of my family members because of whom I am able to complete the degree program successfully.

- Komal Shah

(12MECC25)

List of Abbreviations

2D ..... 2 Dimensional
3D ..... 3 Dimensional
BW ..... Bandwidth
CPU ..... Central Processing Unit
DAQ ..... Data Acquisition System
DDR ..... Double Data Rate
DVFS ..... Dynamic Voltage Frequency Scaling
EIST ..... Enhanced Intel SpeedStep Technology
eMMC ..... Embedded Multi Media Card
FPS ..... Frames Per Second
GFX ..... Graphics
GPS ..... Globle Positioning System
GPU ..... Graphics Processing Unit
HDMI ..... High Density Multimedia Interface
HFM ..... Highest Frequency Mode
HLT ..... Halt
HW ..... Hardware
ID ..... Identity
IDI ..... Improved Data Interchange
IP ..... Intelligent Peripheral
IPC ..... Instruction Per Cycle
JVM ..... Java Virtual Machine
LCD ..... Liquid Crystal Display
LFM ..... Lowest Frequency Mode
LPE ..... Low Power Engine
MB ..... Mega Bytes
Mbps ..... Mega Bytes Per Seconds

MCO ..... Memory and CPU Observer
MSR ..... Model Specific Register
OS ..... Operating System
PA ..... Platform Architect
PC ..... Personal Computer
PLL ..... Phase Lock Loop
SA ..... System Agent
SDIO ..... Secure Digital Input/Output
SOC ..... System On Chip
SRAM ..... Static Random Access Memory
SW ..... Software
TG ..... Task Graph
UI ..... User Interface
VID ..... Voltage Identification
VM ..... Virtual Machine
VP ..... Visual Paradigm
VPB ..... Video Play Back
VPU ..... Virtual Processing Unit
Wi-Di ..... Wireless Display
Wi-Fi ..... Wireless Fidelity

Abstract

Smart-phones, tablets, portable handheld devices are used for prolonged periods without being connected to a power supply. For these devices, end goal is to achieve highest performance with lowest power consumption. So, Energy Efficiency is important in the smart phone world. This project explores few challenges to increase the performance of these devices in order to be competitive by characterizing different workloads. It presents recommendation for Atom LFM. It also explores HW rendering v/s SW rendering for Graphics benchmark. As a case study, the project presents about graphics benchmark characterization with display resolution in order to reach a conclusion of how it affects the performance and which is the best resolution to choose.

Performance and power evaluation play a critical role in the architecture analysis of any platform. Architects needs platform model to check how the new feature or how an optimization will impact on power and performance. Although, there are several simulation approaches for Architectural analysis, starting from Cycle Accurate simulators for detailed modelling to high level Spreadsheets for quick analysis, there remains a significant gap to analyse real world "Use-Cases" at SYSTEM level and appropriately co-design HW/SW for optimal Power/Performance.

This project describes a high level SYSTEM approach to abstract the Use-Case into a set of Task flows running over a topology of IP blocks. Task flow approach eliminates dependency on functional models that are needed for Virtual Platform simulations and enables faster evaluation of Architectural studies. Key trade off in Task flow approach is between ability to run actual application binaries v/s speed with application flow represented by a Task flow based on SW spec. This project presents the Task flow approach for VPB (Video Playback) Use-Case on Android based Intel smart phone and correlation of simulation results with measured to validate the accuracy and correctness of such approach and tool.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Today mobile phones are not only used just as a phone but it is now a smart phone, with more advanced computing capability and connectivity than a regular cellular phone used to make and receive calls. The smart phones includes mail functionality, media players, high pixel digital cameras, video cameras, GPS navigation units, etc. to form one multi-use device. Modern smart phones also include high-resolution touch screens and web browsers that display standard web pages as well as mobile-optimized sites. High-speed data access is provided by Wi-Fi, mobile broadband and bluetooth. With the increase in the use of each feature in the mobile device power consumption increase and performance becomes important.

The embedded systems and various computing devices such as smart phones, tablets, portable handheld web devices, ebook readers are often used for prolonged periods without being connected to a power outlet. This usage pattern poses a new set of challenges related to power and performance. These systems comprise of various modules such as: Camera, Accelerometer, GPS, Orientation sensors, temperature sensors, Radio and Modem chips, CPUs, GPUs and bright LCDs. Also, there are very large variations in the specifications of such devices with varied processors, Wi-Fi chips and other modules, although the use-cases are similar. The challenge is to balance the performance and power so that the device can be used for an acceptable duration within acceptable performance bounds. Performance without power considerations is

meaningless, especially in the smart phone world so energy efficiency is important.

1.1 Project Definition

To characterize workload on Android based Intel Devices. This is basically in two parts.

- Workload energy efficiency characterization on different platforms and explore methodologies to optimize it
- Modelling and simulation of Video Playplay on exsisting platform and correlation with measured results using Platform Architect Tool

1.2 Motivation

Smart phones, tablets and other portable handheld devices are used for long period of time without being charged. This usage pattern poses challenges related to power and performance. The challenge is to balance the performance and power so that the device can be used for an acceptable duration within acceptable performance. End goal is to achieve highest performance with lowest power consumption. So, Energy Efficiency is important in the smart phone world. A very high performance without power consideration is meaningless in case of portable device. It is the energy which needs to be efficient. Also, mobile platform needs to be optimized and upgraded, to be with the competition. It becomes necessary to characterize workload before the silicon is ready so that hardware changes can be done before the silicon and this helps to save cost. It is necessary to study the characterization of workload on the next generation platform before the actual silicon. A need arises for a SYSTEM level approach and tool which supports it to model and simulate the real life use-case. This project studies workloads energy efficiency characterization and different methodologies/scopes to optimize it in terms of enery efficiency. It also presents a Task Flow

approach for system level modelling of use-case. Simulation of existing platform on a platform architect tool and correlation with the silicon measurement for Video Playback use-case is also included.

1.3 Organization of the Report

This report is organized in six chapters, including present chapter. First chapter includes project definition, motivation, scope of work and organization of report. Second chapter is the literature survey, which briefs about the basics of command line approach to explore android device, Android architecture, Power and Performance trade-off, CPU Power states, etc., that forms the basics of all workload characterization. Chapter 3 is about the Power, Performance and Characterization measurement for any workload on Android devices and a case study of LFM recommendation for the future platform. Chapter 4 describes about graphics. Different types of rendering and reason for choosing graphics benchmark for characterization. It includes 2 case-studies. One is on HW rendering v/s SW rendering and other is on impact of display resolution on performance. Chapter 5 is about the SYSTEM Task Flow approach and Platform Architect tool. It explains the function of tool with an example. Also, it includes building model for existing platform and simulating for Video Playback workload. At the end correlation of simulation results with silicon measurement is presented. Conclusion of the thesis and Future scope of the work is described in Chapter 6.

Chapter 2

Literature Survey

This chapter starts with exploring the android device, a command line approach to know android device and to change few parameters like frequency with which CPU is working, governors, number of cores, etc. It then presents Android Architecture and platform hardware - a view of Intel's smart phone SOC (System On Chip).

To evaluate the performance of Android devices and to compare different platforms various benchmarks (workloads) are available in the market. It becomes necessary to characterize these benchmarks in terms of CPU Utilization, CPU Frequency, GPU Utilization, GPU Frequency, Memory Bandwidth, Core scaling, Display Resolution, etc.

In order to understand the behaviour of any benchmark on a platform it is necessary to understand CPU states. For a high intensive workload, CPU might get fully utilized at maximum frequency. Similarly, a low profile workload utilizes lower CPU frequency for less fraction of time. This also affects the performance of the workload and together energy efficiency of the workload. So, literature survey dives into details of CPU Power and Performance States and presents trade-off between Power and Performance.

2.1 Exploring CPU of the Android Device Using Command Line

To explore CPU of any android device, it is necessary to have root access of the device. All Intel's device are by default as root user for easy testing and troubleshooting.

2.1.1 General Commands to Explore CPU

For memory information

```
cat /proc/meminfo
```

For CPU information

```
cat /proc/cpuinfo
```

To find no of cores

```
cat /proc/cpuinfo | grep processor | wc l
```

To find Max frequency

```
cat /sys/devices/system/cpu/cpu0/cpufreq/cpuinfo_max_freq
```

To find Min frequency

```
cat /sys/devices/system/cpu/cpu0/cpufreq/cpuinfo_min_freq
```

To find Current working frequency

```
cat /sys/devices/system/cpu/cpu0/cpufreq/scaling_cur_freq
```

To find the supported frequency of the device

```
cat /sys/devices/system/cpu/cpu0/cpufreq/stats/time_in_state
```

or

```
cat /sys/devices/system/cpu/cpu0/cpufreq/scaling_available_frequency
```

To disable hyperthreading/core

```
echo 0 > /sys/devices/system/cpu/cpu1/online
```

To enable hyperthreading

```
echo 1 > /sys/devices/system/cpu/cpu1/online
```

Enter frequency in kHz. Range of frequency can be set by giving different frequency

to maximum and minimum.

2.1.2 Governors [1]

Governors estimate the load on each CPU. They run an algorithm to decide as to when the frequency of the CPU has to changed. Information about the governors can be found using

`echo/sys/devices/system/cpu/cpu0/cpufreq/scaling_available_governors`

Output: ondemand userspace interactive performance

To change governor

`cat performance > /sys/devices/system/cpu/cpu0/cpufreq/ scaling_governor`

Ondemand

This governor scales CPU depending on the need basis on the fly.

Userspace

In this governor user can specify/fix minimum & maximum frequency of his choice between $(P_0 - P_n)$.

Interactive

Similar to ondemand governor, this is more aggressive. It has considerably less delays in scaling up the frequency.

Performance

This governor always locks frequency to maximum i.e. P_0 always, even though idle.

2.2 Android Architecture

Android is a Linux based operating system, designed primarily for mobile devices such as smart phones and tablet computers. Android is open source. It allows the software to be freely modified and distributed by device manufacturers and developers. The Android OS is referred as a software stack of different layers, where each layer is a group of several program components.

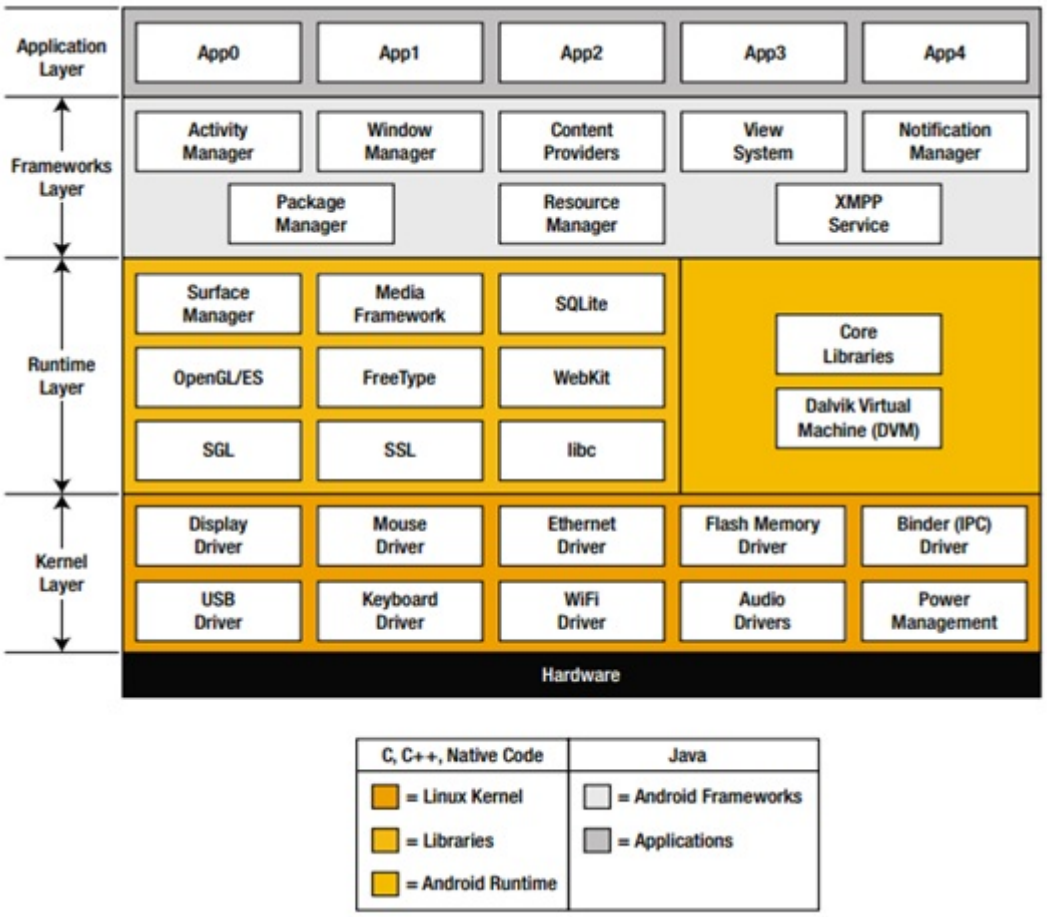

Figure 2.1: Android Architecture [2]

The basic layer is the Linux kernel. It is this Linux layer that interacts with the hardware and contains all the essential hardware drivers. Drivers are programs that control and communicate with the hardware. The Linux kernel also acts as an abstraction layer between the hardware and other software layers. Android uses the Linux for all its core functionality such as memory management, process management, networking, security settings etc.

The next layer is the Androids native libraries. It is this layer that enables the device to handle different types of data. These libraries are written in c or c++ language and are specific for a particular hardware.

Surface Manager is used for compositing window manager with off-screen buffering. Off-screen buffering means you cant directly draw into the screen, but your drawings go to the off-screen buffer. There it is combined with other drawings and form the final screen the user will see. This off screen buffer is the reason behind the transparency of windows [2].

In the runtime layer there are two sections, Libraries and Android Runtime. The libraries component shares its space with the run-time component. The libraries component acts as a translation layer between the kernel and the application framework. The libraries are written in C/C++ but are exposed to developers through a Java API. Developers can use the Java application framework to access the underlying core C/C++ libraries. Some of the core libraries include the following [2]:

- **LibWebCore:** Allows access to the web browser.
- **Media libraries:** Allows access to popular audio and video recording and playback functions.
- **Graphics libraries:** Allows access to 2D and 3D graphics drawing engines.

Android Runtime consists of Dalvik Virtual machine and Core Java libraries. Virtual Machine is an isolated, guest operating system running within another host operating system. This execute applications as if they were running on a physical machine. One of the main advantages of a virtual machine is portability. Regardless of the underlying hardware, the code will work on the VM. It is possible to write code only once and execute it on any hardware platform that runs a compatible VM [2]. It is a type of JVM used in android devices to run apps and is optimized for low processing power and low memory environments. Unlike the JVM, the Dalvik Virtual Machine doesnt run .class files, instead it runs .dex files. .dex files are built from .class file at the time of compilation and provide higher efficiency in low resource environments.

Application Framework is the blocks that interact with the application directly. These programs manage the basic functions of phone like resource management, voice call management etc.

- **Activity Manager:** Manages the activity life cycle of applications
- **Content Providers:** Manage the data sharing between applications
- **Telephony Manager:** Manages all voice calls. Used to access voice calls in application.
- **Location Manager:** Location management, using GPS or cell tower
- **Resource Manager:** Manage the various types of resources in application

Applications are the top layer in the android architecture and this is where our applications are going to fit. The application component of the Android operating system is the closest to the end user. This is where the contacts, phone, messaging, and angry birds apps live.

2.3 Mobile Platform

A platform is an integrated set of processor, a lightweight form-factor for mobility, technologies to provide great battery life, wireless capabilities and applications that use the mobile form-factor efficiently. The final product is the sum of a set of parts that provides the valuable end user experience - it is not just a device, but a platform [3].
The five major ingredients in a platform are:

- **Hardware:** Such as the processor, the chipset, the modem, memory, etc.
- **Software:** Operating Systems, Applications, Firmware and Compilers
- **Technologies:** Intel Hyper-Threading Technology, Intel SpeedStep Technology, etc.
- **Standards and Initiatives:** Such as Wi-Fi, Wi-Max, etc.

- **Services :** such as digital media distribution, communications services and system services

In general, a platform is an underlying computer system on which application programs can run. One such mobile platform of Intel is as shown in Figure. 2.2.

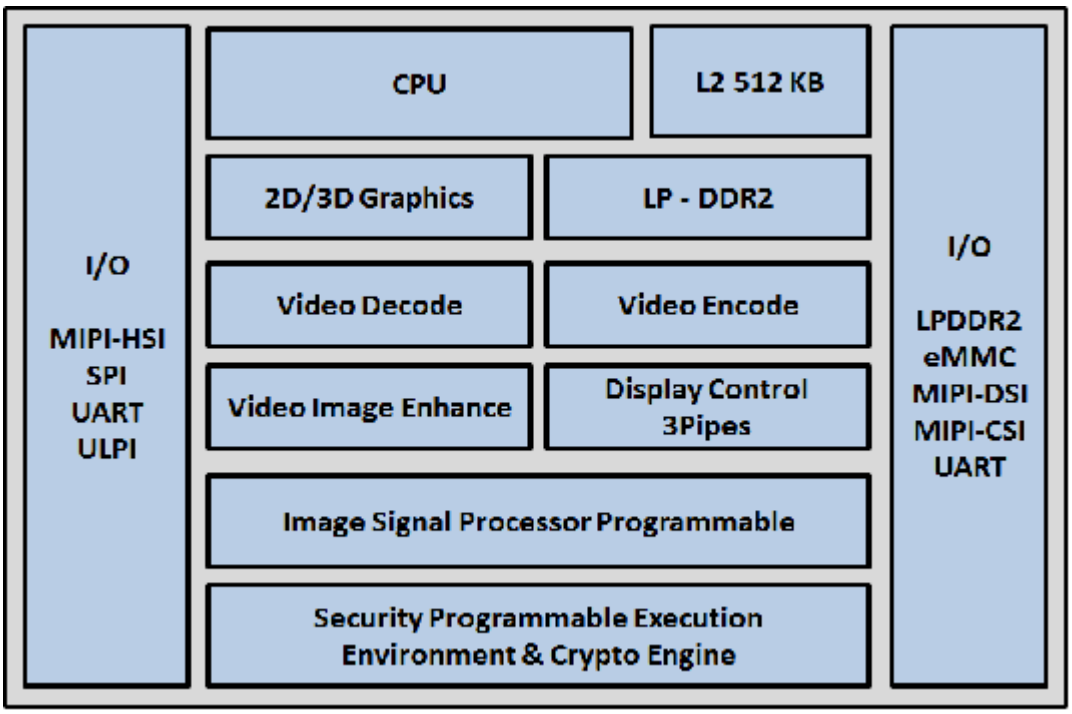

Figure 2.2: CloverTrail Plus SOC [4]

Mobile Platfoms are designed specifically for mobile devices such as smart phones. CloverTrail Plus is Intels first dual core Smart Phone Platform with 32 nm CPU core running at 2 GHz. Medfield is the platform, Penwell is the SOC and the CPU inside Penwell is codenamed as Saltwell.

2.4 CPU Power Management

Power management is essential for any portable device due to three reasons.

a. Longer Battery Life

b. Thermal Management : Proper Skin Temperature

c. Lighter Device

In mobile devices running real time applications, the power consumed by the CPU can be more than the needed amount of power. Power management is concerned with minimizing the power consumed by a system. To manage power, it is vital to understand how the power is consumed by different system components. The power management strategy should focus on the major components that consume power to reduce the overall system power consumption. Figure. 2.3 shows a typical power consumption distribution for a handheld device.

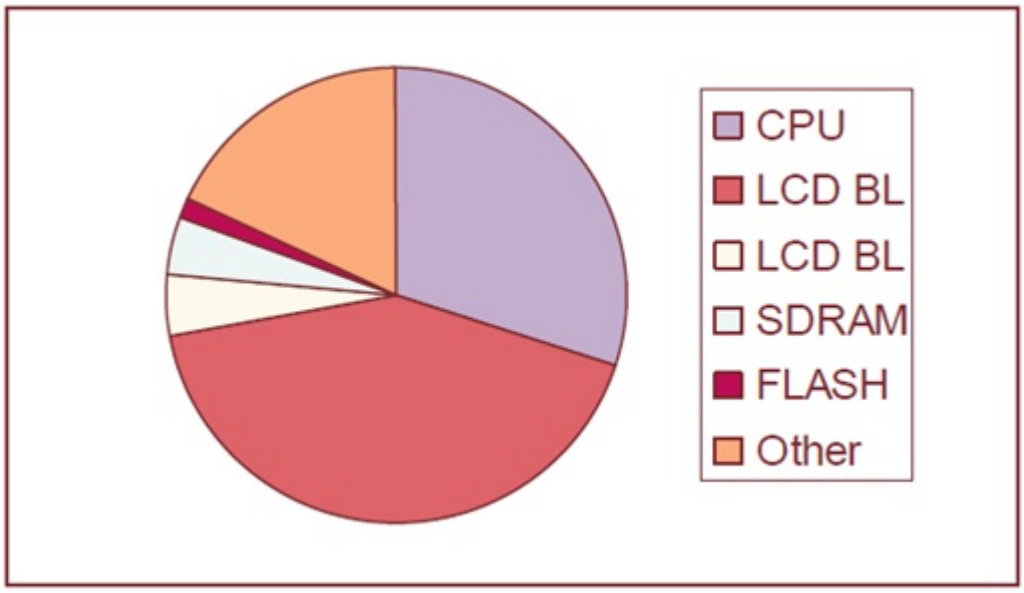

Figure 2.3: Power Distributions for Typical Handheld Device [5]

Maximum power is consumed by LCD BL (LCD Back-Light). But scope of optimization is very less as it completely depends on screen size and brightness. Next most consuming power component is CPU. Power consumption in CPU depends on workload, operating frequency, governor, computing time, dispaly resolution. Today a

seperate HW GPU is included for hard computing like 2D and 3D computing. Hence there are lot of scope of power optimization in CPU and GPU. In this project, most of the focus is on CPU and GPU optimization.

2.4.1 Power and Performance Trade-Off

The requirement for any handheld device is, high performance with low power consumption.
Generally, dynamic power consumption is given by [5]

$$P = NCV^2f \tag{2.1}$$

C : CMOS circuit output load capacitance
N : Average number of switching activities for clock cycle
V : Supply voltage
f : Clock frequency
This shows that power consumption is directly proportional to clock frequency. As clock frequency increases performance improves but from above formula, power consumption will also increase. Thus there is always a trade-off between power and performance. End goal is to have such point of frequency and voltage so that power and performance are under acceptable level and with the competition.

2.4.2 Enhanced Intel SpeedStep Technology

Enhanced Intel SpeedStep Technology (EIST) allows the system to dynamically adjust processor voltage and core frequency, which results in decreased power consumption, which results in decreased heat production. It introduces a means of enabling high performance while meeting the power-saving needs. Large power saving is possible if frequency and voltage are reduce at the same time. It throttles clock speed as per demand of load thus save power yet achieve high performance.

Enhanced Intel SpeedStep == Dynamic Frequency and Voltage scaling

The following are the key features of Enhanced Intel SpeedStep Technology [6].

- Multiple voltage and frequency operating points providing optimal performance at the lowest power
- Voltage and frequency selection is software controlled by writing to processor MSRs
  - If the target frequency is higher than the current frequency, V_{CC} is ramped up in steps by placing new values on the VID pins and the PLL then locks to the new frequency.
  - If the target frequency is lower than the current frequency, the PLL locks to the new frequency and the V_{CC} is changed through the VID pin mechanism.
  - Software transitions are accepted at any time. If a previous transition is in progress, the new transition is deferred until the previous transition completes.
- Improved Intel Thermal Monitor Mode
  - When the thermal sensor indicates that the temperature is too high, the processor can automatically perform a transition to a lower frequency and voltage specified in a software programmable MSR.
  - The processor waits for a fixed time period. If the temperature is down to acceptable levels, an up transition to the previous frequency and voltage point occurs.
  - An interrupt is generated for the up and down Intel Thermal Monitor transitions enabling better system level thermal management.
- Enhanced Thermal Management Features

- Digital Thermal Sensor and out of specification detection.
- Intel Thermal Monitor 1 (TM_1) in addition to Intel Thermal Monitor 2 (TM_2) in case of unsuccessful TM_2 transition.

2.4.3 CPU States

CPU consumes a significant amount of power and so, it should be considered for power management. This gave rise to CPU States.

2.4.3.1 C States

In order to save energy when CPU is idle, the CPU can be commanded to enter a low-power mode. Each CPU has several power modes and they are collectively called "C States".
The basic idea of these modes is to cut the clock signal and power from idle units inside the CPU. The more units are stopped or partially stopped by completely shutdown or reducing voltage respectively, more energy is saved, but time required for the CPU to wake up and become 100% operational is more.
These modes are known as C States. They are numbered starting at C_0, which is the normal CPU operating mode, i.e. CPU is 100% turned on. The higher the C number is, deeper is the CPU sleep mode, i.e. more circuits and signals are turned off and more time the CPU will take to go back to C_0. In the Table I [7, 8], all C States are summarized.

C_1 State [7, 8]
All x86 CPUs have an instruction called "HLT", where the CPU remains idle, doing nothing when it ran. CPU becomes operative when it receives an interrupt. Since in halt mode, CPU is completely idle, Intel added the Auto Halt mode known as C_1 state. When HLT instruction is executed, CPU enter halt mode and internal CPU clock signal is stopped. Only two units inside CPU remain on. They are Bus Interface and the Interrupt Controller. This is done to allow the CPU to wake up if request

State	Name	CPU Internal Clock	Bus Interface/ Interrupt Controller	L1 Cache	L2 Cache	CPU Voltage	Wake up Time
C_0	Operating State	On	On	-	-	-	Less
C_1	Halt	Stop via SW	On	-	-	-	More than C_0
C_1E	Enhanced Halt	Stop via SW	On	-	-	Reduced	More than C_0
C_2	Stop Clock	Stop via HW	On	-	-	-	More than C_1
C_2E	Extended Stop Clock	Stop via HW	On	-	-	Reduced-Less than C_2	More than C_1
C_3	Sleep	Stops all	Stops all	-	-	-	More than C_2
C_4	Deeper Sleep	Stops all	Stops all	Flushed	Partial Flush	Reduced-Less than C_3	More than C_3
C_4E/C_5	Enhanced Deeper Sleep	Stops all	Stops all	Off	Flushed	Reduced-Less than C_4	More than C_4
C_6	Deep Power Down	Stops all	Stops all	Off	Off	Reduced-Less than C_5	More than C_5

Table I: CPU - C-States

comes through the CPU external bus. As soon as CPU receives an interrupt signal it goes to C0 State with clock signal being restored.

C_2 State [7, 8]

C_2 State was introduced by adding a extra pin to the CPU called "STPCL" (Stop Clock). When it is activated, CPU core clock is stopped. This is same as C_1 state with only difference in how they achieve the state. C_1 state is activated by software while C_2 is activated by hardware i.e. by sending a signal to CPU pin "STPCLK". Here also, Bus Interface and the Interrupt Controller remain on to serve important request. Some CPU have enhanced Halt/Stop state i.e. C_1E/C_2E, which reduces the CPU voltage to save power.

C_3 State [7, 8]

C_3 is called Sleep state. It cuts all internal clock signals from CPU including bus interface unit and interrupt controller. This means that when CPU is in sleep mode it cannot answer to important request coming from CPU external bus or interruption.

C_4 State [7, 8]

Modes C_1, C_2 and C_3 deals basically with the clock signal. Since in C_3 state all clocks are stopped there is no way to save power controlling clock signal. The next step is to control CPU voltage. By reducing CPU voltage power can be saved since power consumption is directly proportional to square of voltage. C_4 is called Deeper sleep state where L2 cache is partially flushed and voltage is reduced to save power. When CPU is waked up it loads the previous state of all internal units from static RAM. Waking up the CPU from this state takes longer time than the previous states.

C_6 State [7, 8]

When CPU enters C_6 state it saves its entire architectural state inside a special static RAM, which is fed from an independent power source. This allows CPU internal volt-

age to be lowered to any value, which completely turn off the CPU when it is idle. When CPU is waked up it loads the previous state of all internal units from static RAM. Waking up the CPU from this state takes longer time than the previous states.

2.4.3.2 P States

P States are called performance states of CPU. P-state is both a frequency and voltage operating point. Both are scaled as the P-state increases. When a core is in the C_0 state, it can attain one of the performance states $P_0 - P_n$ [7, 9].

P_0: Highest performance/frequency also called HFM
P_1: Intermediate frequency
P_2:
...
P_n: Lowest frequency also called LFM

Performance is directly related to frequency. As operating frequency increases, performance of processor increases. The same applies for decreasing the frequency. If frequency is made half, a compute bound task runs half as fast. For example, if a task is compute bound and requires 100% of the CPU for 1 second at 2 GHz, it will take 2 seconds to execute at 1 GHz. (This is roughly correct. There are other factors influencing runtime, such as cache size and speed, interrupts, etc.)
If frequency is decreased, CPU utilization increases and % idle time reduces. This shouldn't have any effect on the power usage of the processor. Here voltage scaling comes into play.
There are two primary reasons for P-states, one is to reduce the peak thermal load, and the other is to save power.

Reducing Peak Thermal Load

The instantaneous energy usage (power) of the processor is related to its activity. If

the processor is very busy, it runs hotter. So, reducing the frequency reduces the peak thermal output even if the total energy usage is not reduced. The advantage of reducing peak thermal load has also to do with maximum skin temperature. So, if peak power is reduced; the maximum skin temperature is reduced which directly impact user experience [7, 9].

Chapter 3

Power and Performance Measurement

Measurement is the first step for any optimization. It shows how different workloads behave on different platforms and where optimization can be done. There are three basic and most important things for measurement.

- Power
- Performance
- Characterization of Workload

The power tool helps to measure power consumed by different platform components for specific workload by looking at the various power rails going to the components. The performance measurement of different benchmarks which are basically Android Applications running on the platform depends on the type of Workload. For browsing case time is important. Lesser the time better the performance. Whereas for workload such as games, FPS is considered for performance measurement. Measurement done for characterization of any workload is basically for debugging. It is used to evaluate the performance and utilization of the different components of the system viz. CPU, GPU, Memory, etc.

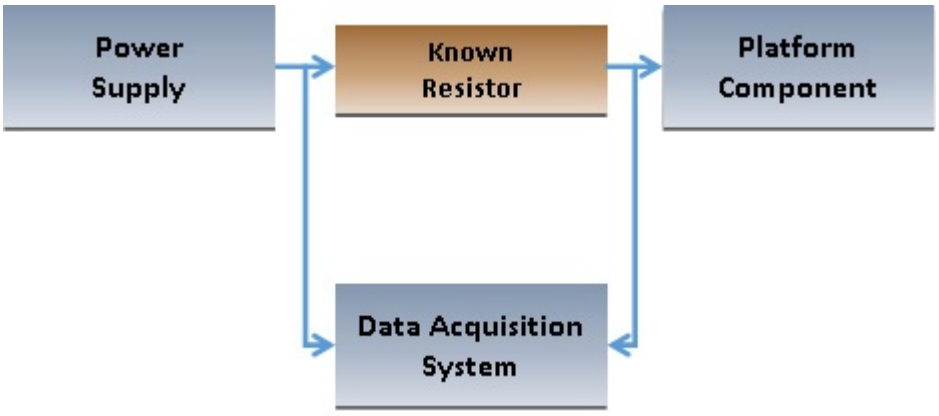

Figure 3.1: Power Measurement Setup

This chapter ends with an experiment for Atom LFM. By characterizing and measuring energy for different workloads options/optimizations is suggested for next generation platforms. This experiment recommends for Atom LFM.

3.1 Power Measurement

A computer platform consists of many components viz., CPU, Memory, Display etc. These behave differently and consume different amount of power for various workloads. Understanding the impact on various platform components because of workloads is necessary. Thus power measurement is an important aspect for testing and optimizing a platform. Usually DAQs (Data Acquisition Systems) are being used for power measurements.

DAQ are basically used for two reasons:

a. For accurate power measure

b. To get the result in PC

The basic concept is as follows (Refer Figure. 3.1).

- A sense resistor is connected in series with the component whose power consumption is to be calculated. Power rails from sense resistor are given to the

DAQ.

- DAQ measures two voltages.

  a. Differential voltage across the sense resistor (dV)

  b. Voltage at point B with respect to ground (V)

- Differential voltage is sampled at regular interval and is divided with the value of the sense resistor to get the current (I) through the rail. Samples are taken for accuracy. Smaller the interval, better the results.

- Computed current (I) is multiplied with absolute rail voltage (V) to get the instantaneous power of the component and average is taken over all the samples for a specific period.

- P = VI gives the input power. Similarly output power is calculated and there difference gives power consumed by the component.

3.2 Performance Measurement

Performance measurement is way to evaluate devices. There are various ways to measure performance. Performance can be in terms of time or FPS depending on the type of workload and what needs to be evaluated. For example,

- For graphics benchmark FPS or score helps to evaluate/compare devices. Each benchmark calculates score using FPS of the test running which is displayed at the end of the test like AnTuTu, Quadrant 2D, etc. Some benchmark even gives FPS number also like GLBenchmark, BaseMark X, etc.

- For games workload time and FPS, both matters depending on what is the focus on. If the focus is on application launch then time is important and if the focus is on the display and game experience then FPS is important.

- For browsing benchmarks, how fast the page load is important so, time is measured to evaluate performance of such benchmarks.
- For CPU intensive workload/benchmark like Core Mark, Caffeine Mark, etc. time taken to complete the benchmark evaluates the CPU performance. So, time is measured for performance.

To evaluate the performance of any device choosing proper benchmark is very important. Benchmark should depict the real workloads in the platforms. The benchmarks instead of real use-case are usually choosen to have repeatability.

3.3 Characterization Measurement

Any measurement that is done for characterization of any workload is basically for debugging purpose like reason for low FPS, effect of increasing CPU frequency, utilization of the different components of the system viz. CPU, GPU, Memory, etc. Various internal/external tools are used to measure such characteristics.
There are three mechanisms to capture residencies/utilization.

- **Snapshot :** This collects data at start and end of the specified time.
  Features such as C State, south complex residency are measured using snapshot. This has minimum overhead.

- **Polling:** This collects data at regular interval for specified time.
  Most of the feature residencies are calculated using this method. Features such

as Graphics C State, Graphics P State, Memory BW, Temperature, etc. are measured using polling. This has maximum overhead.

- **Tracing:** This collects data only at the change between specified time.
  Features such as wakelock, interrupts request are measured using tracing. For

this overhead depends on number of times it has to measure.

The overhead for all the mechanics are within the acceptable limits, it affects only 2 - 3% of the result.
Power and performance measurement together called energy is important for evaluating any platform. Performance/watt is important for a particular platform. Characteristic measurement helps debugging and understanding the workload behavior.

3.4 Case Study: Recommendation for Atom LFM

This case study presents an analysis on CPU frequency scalability for different workloads on Intels upcoming Platform for Android devices. Workloads/Benchmarks are set of android applicationsthat are formed to stress different parts of the system. This chapter uses basic benchmark/workload of Android devices like video playback, audio playback, etc. to observe frequency scalability.

Objectives

- Present an analysis on CPU Frequency scalability for different workloads
- Recommendation to Atom CPU about LFM

3.4.1 CPU Frequency Scalability

CPU runs at different frequency depending on the workload to save power. Each platform has some lowest and highest frequency supported. Workload utilizes these range of frequency based on the requirement to give better performance at minimum power consumption. An analysis is presented on CPU utilization and power consumed for different CPU frequency for different workloads.

3.4.1.1 Video Playback

A real world workload of video playback was chosen first as this is the most used load on any android device. Here a video of 1080 pixel resolution at 30 FPS was chosen for test. Intel has already defined particular videos of different resolution and FPS for comparing across platform. Same video was chosen for test. Here platform is multi-core but for better understanding of scalability, platform is converted to single core and single thread by switching off other core and multi-threading property. The trend remains same for single core and multi-core. Figure. 3.2 shows CPU utilization for core 0 and core 1 with frequency of CPU.

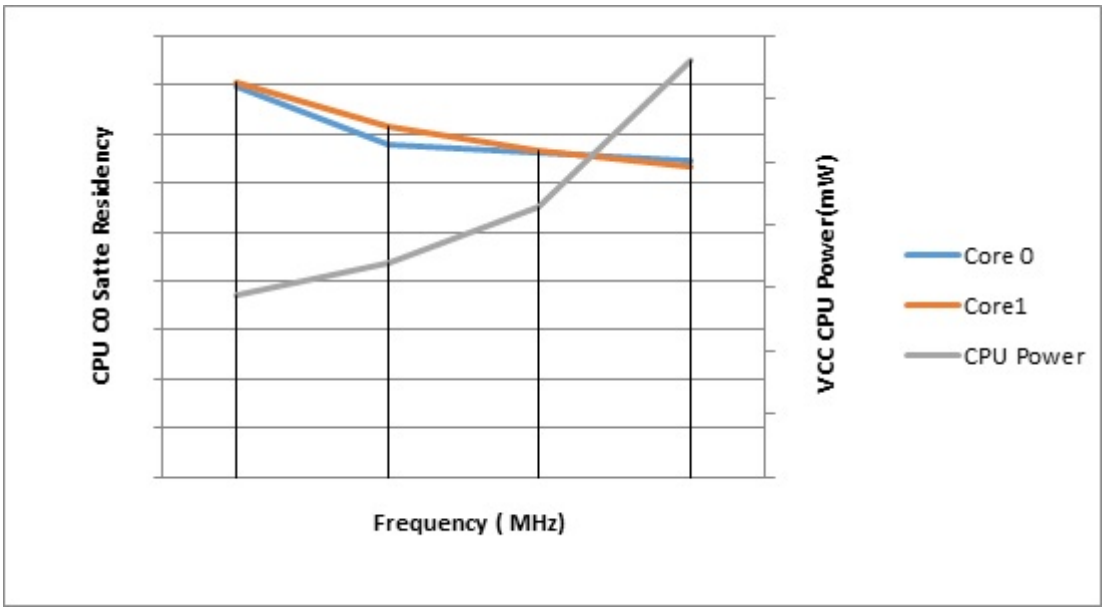

Figure 3.2: Video Playback - Dual Core Frequency Scaling

Figure. 3.3 shows same with core 1 off. It can be seen that trend of the curve for both is same.As the trend is same, only single core results are used for better understanding.

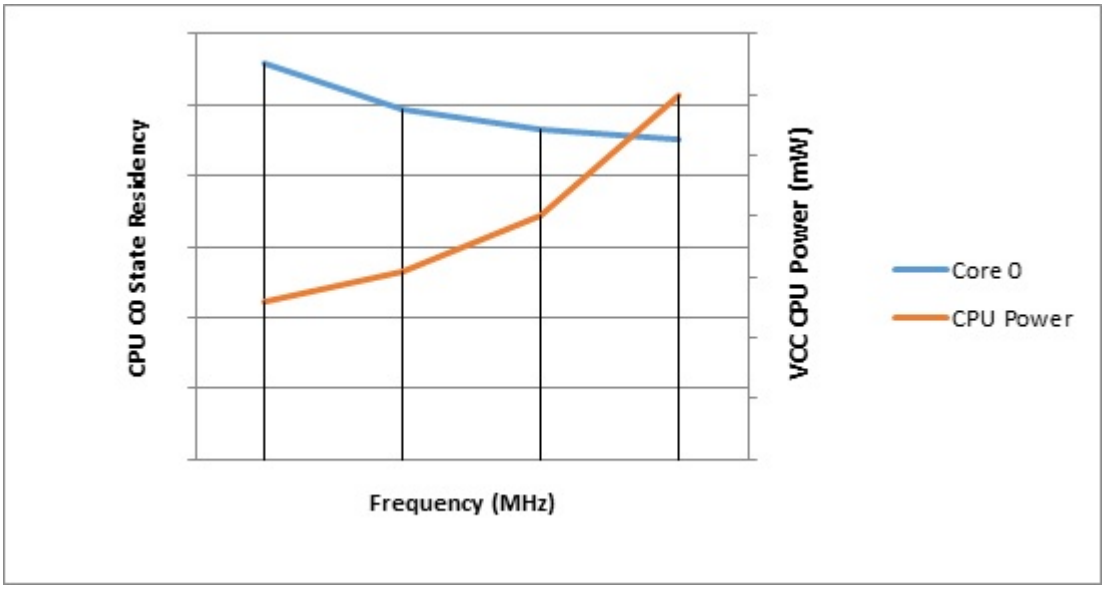

Figure 3.3: Video Playback - Single Core Frequency Scaling

Inference

CPU utilization decreases with increasing frequency but scaling drops post second frequency point. Scaling saturates with increasing frequency. CPU power increases with frequency. Power delta between lowest and second lowest is less and that is due to voltage increase. For Video playback no significant difference for LFM at 1^{st} or 2^{nd} frequency point.

3.4.1.2 Video Record

Video record is the other workload which is widely used in mobile world. For experiment, video is recorded with 1080 pixel resolution set and measurements are done. Figure. 3.4 shows the scaling of CPU utilization and power with different operating frequency of CPU.

The FPS of recorded video was measured and it comes to be 30 FPS, which is good.

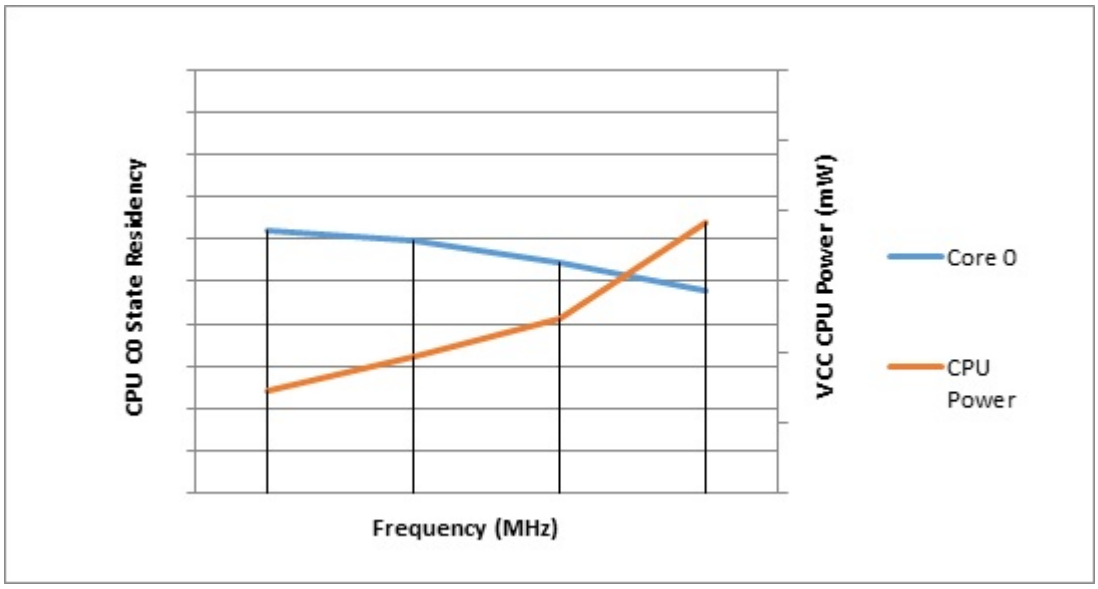

Figure 3.4: Video Record - Single Core Frequency Scaling

Inference

CPU utilization decreases with increasing frequency. But decrease for lowest and second lowest frequency is much less. CPU power increases with frequency. Power delta between lowest and second lowest is less due to voltage increase. Same as Video playback, for HW accelerated workload there is no higher frequency LFM gain or loss. Depending on other workload like graphics, LFM at 1^{st} frequency or 2^{nd} frequency point may help to achieve energy efficiency gain.

3.4.1.3 Basemark Taiji

Basemark Taiji is a graphics benchmark. This characterizes the graphics of the system.

Inference

CPU Utilization scales with frequency up to second frequency point. Performance is 24 FPS for all frequency. May be gain in power as CPU utilization is significantly less.

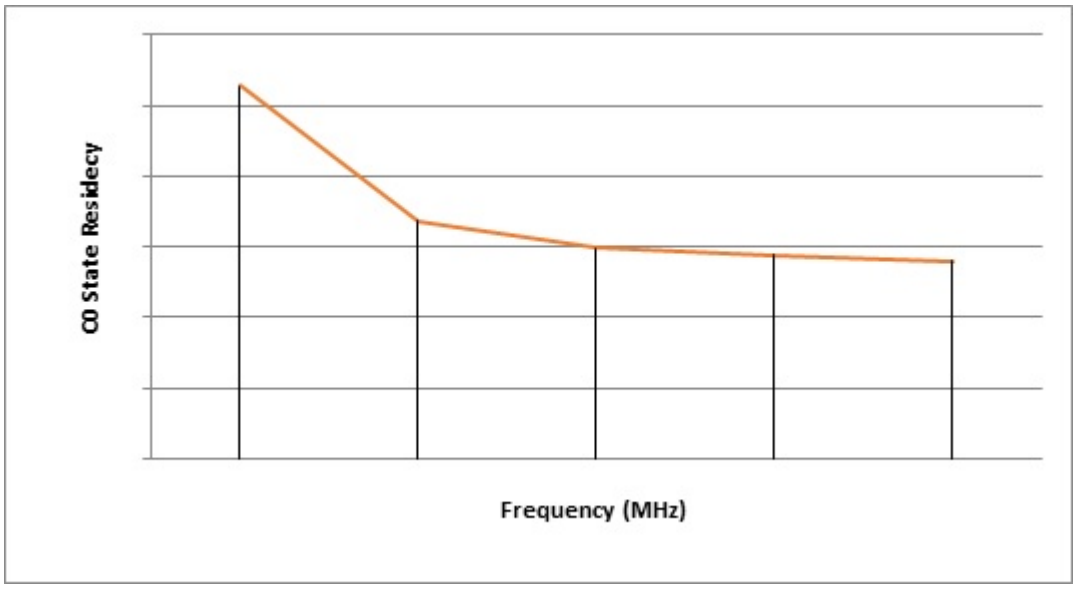

Figure 3.5: Basemark Taiji - Single Core Frequency Scaling

3.5 Conclusion

Workload scales non-linearly. Higher CPU frequency doesn't reduce C_0 residency in step. Workload scales nicely till second operating frequency. CPU Utilization does not scale beyond this knee point. Power increases with frequency due to voltage scaling. If power gain is obtained for Graphics workload (Taiji), recommend "Second Frequency" point as optimal point. Scalability determines the most energy efficient LFM even with higher voltage.

Chapter 4

Case Study: Android Graphics

Graphics intensive computer games are no longer restricted to high performance desktops, but are also available on a variety of portable devices ranging from notebooks to tablets and mobile phones. Battery life has been a major concern in the design of both the hardware and the software for such devices. Energy efficiency is one of the most critical issues in the design of such battery-powered portable devices. Motivated by the below mention facts this chapter attempts to characterize various games and graphics benchmarks.

- Increasing availability of game applications in Android world
- All unit of platforms are utilized like display is 100% ON, GPU and CPU
- It is most interactive. Accelerometer and touch are used.

Due to all these reasons, this is the most energy constraint and so, characterization become important.
There are two fundamental differences between the workloads arising from game and video decoding applications [10]:

- The magnitude of the variation in the number of processor cycles required to process a frame is significantly higher in case of games compared to video decoding. However, the frequency of this variation is much higher in the case

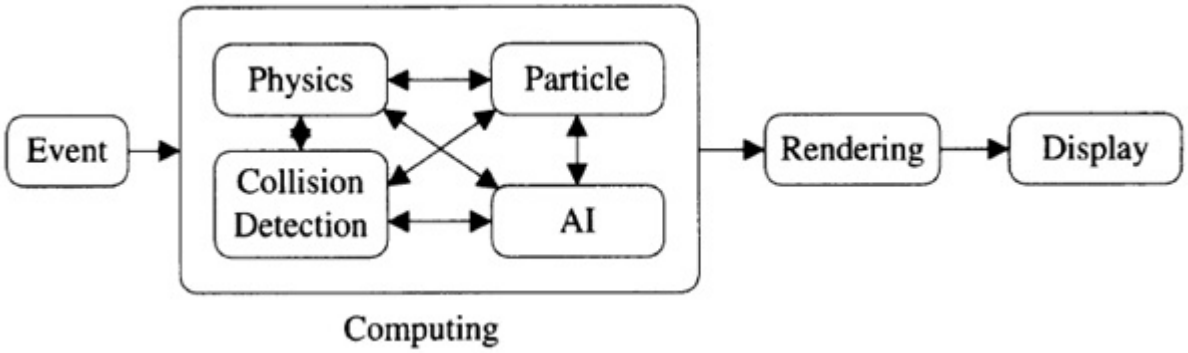

Figure 4.1: Frame Processing in Game Application

of video decoding applications. In other words, compared to workload arising from a game, a video decoding workload exhibits a smaller but more rapid variation. This observation indicates that the potential energy savings that may be obtained from applying DVFS to games is higher than what may be obtained from video decoding applications.

- In the case of game applications, the frames contain structure, which can be exploited to predict their workload or processor cycle requirements. While processing a frame, the workload depends heavily on the scene that the frame is depicting. More specifically, the workload depends on the content of the frame or the constituting objects that need to be processed. In contrast, video frames offer much less structure.

Before going to actual game/graphics benchmark characterization, how actually frame is processed in a game application and various ways to render a frame is discussed.

4.1 Anatomy of a Game Engine [10]

A game engine runs in an infinite loop, where the body of this loop consists of task responsible for processing a single frame. This loop body is shown in Figure. 4.1 Event denotes the user inputs or interactions with the game, which along with the current state of the game is used to generate the next frame to be displayed. This

involves two sequential steps, computing and rendering which describe below.

Computing

The computing step comprises tasks such as collision detection, AI, simulation of game physics and particle systems. Collision detection includes algorithms for checking collisions between the different objects and characters in the game. Such algorithms compute intersections between two given solids, their trajectories as they move, impact times during a collision and their impact points. In some engines, the AI tasks determine the movement of the characters in the game. Game physics incorporates physical laws into the game engine so that different effects (e.g. collisions) appear more realistic to a player. Typically, simulation physics is only a close approximation ofreal physics, and computation is performed using discrete rather than continuous values. Finally, a particle system model allows a variety of other physical phenomenon to be simulated. These include smoke, moving water, blood, explosions and gun fires. The numbers of particles that may be simulated are typically restricted by the computing power of the machine on which the game is being played.

The rendering step involves algorithms to generate an image (or a frame) from a model, which is then displayed as shown in Figure. 4.1. In this case, the model is typically a description of several three dimensional objects using a predefined language or data structure. It consists of geometry, viewpoint, texture and lighting information. In the case of 3D graphics, rendering may be done offline, as in pre-rendering, or in real time. Pre-rendering is a computationally intensive process that is typically used for movie creation, while real-time rendering is commonly done in 3D computer games, which often rely on the use of a specialized processor called a Graphics Processing Unit (GPU).

The rendering steps include the transformation of the vertices of solid objects to the screen space, deletion of invisible pixels by clipping, rasterization, and interpolation of various parameters. The outcome of these steps is the transformation of the 3D data onto the 2D screen. Rendering is computationally expensive and occupies a significant fraction of the total processing time of a frame.

4.2 Hardware Acceleration

Hardware acceleration is the use of computer hardwareto perform some functions in computing faster than possible than running software on the CPU.
Generally, processors are sequential, and instructions are executed one by one. Various techniques are used to improve performance; hardware acceleration is one of them. The main difference between hardware and software is concurrency, allowing hardware to be much faster than software. Hardware accelerators are designed for computationally intensive software code.
The hardware that performs the computation and is responsible for accelerating the task is a separate unit from the CPU. It is referred as a hardware accelerator, or GPU (Graphics Processing Unit).
The 2D operations are done on GPU instead of CPU. 2D operations are translated into equivalent 3D operations and processed in GPU. The performance will be faster but GPU power will be more compared to software rendering.

4.3 Software Rendering

Software rendering refers to a rendering process in absent of graphics hardware, such as a GPU. The rendering takes place entirely in the CPU. Rendering everything with the CPU has the main advantage that it is not limited to the capabilities of graphics hardware, but the disadvantage is that it gives lower performance and speed. Software rendering can be split into two main categories:

a. Real Time Rendering or Online Rendering

b. Pre-rendering or Offline Rendering

Real-time rendering is used to render an interactive scene, like in games, and generally each frame must be rendered in a few miliseconds. Offline rendering is used to create images and movies, where each frame can take hours or days to complete.

Real Time Software Rendering - Online Rendering

In real-time rendering, all the images from 3D geometry, textures, etc. are produced on the fly and displayed to the user as fast as possible. The user can interact with the 3D scene using a variety of input devices.

Offline rendering

Offline rendering refers to anything where the frames are rendered to an image format, and the images are displayed later either as a still, or a sequence of images. Images are rendered on a frame buffer and when frame is ready it is displayed as an image.

4.4 Case Study: Quadrant 2D - Analysis of HW/ SW Rendering

The Quadrant benchmark application for Android runs a series of processor intensive tests to determine the performance of Android device for comparison to other Android devices. The purpose of a benchmark is to provide a consistent standard by which things can be measured. Quadrant runs a total of 21 tests covering the processor, memory, input, output, 2D graphics and 3D graphics performance.
Quadrant 2D test is primarily for testing 2D graphics performance of a phone/device. 2D graphics is used in case of basic UI of android and games like angry birds etc.
As discussed in section 4.2 and 4.3 there are two ways to render 2D graphics on an android system. One is HW accelerated and second is through SW library i.e. on CPU.

a. **SW Rendering:** For using CPU for 2D rendering, there is library called libskia.so, which is being used and there is more load on CPU than on GPU. So VCC power (CPU power) would be more. At the same time performance of the test is not so good i.e. not reaching 60 FPS [or score of 1000] due to limitation on CPU for doing these operations. So, in this case CPU utilization will be high, GPU utilization will be low and performance will be low.

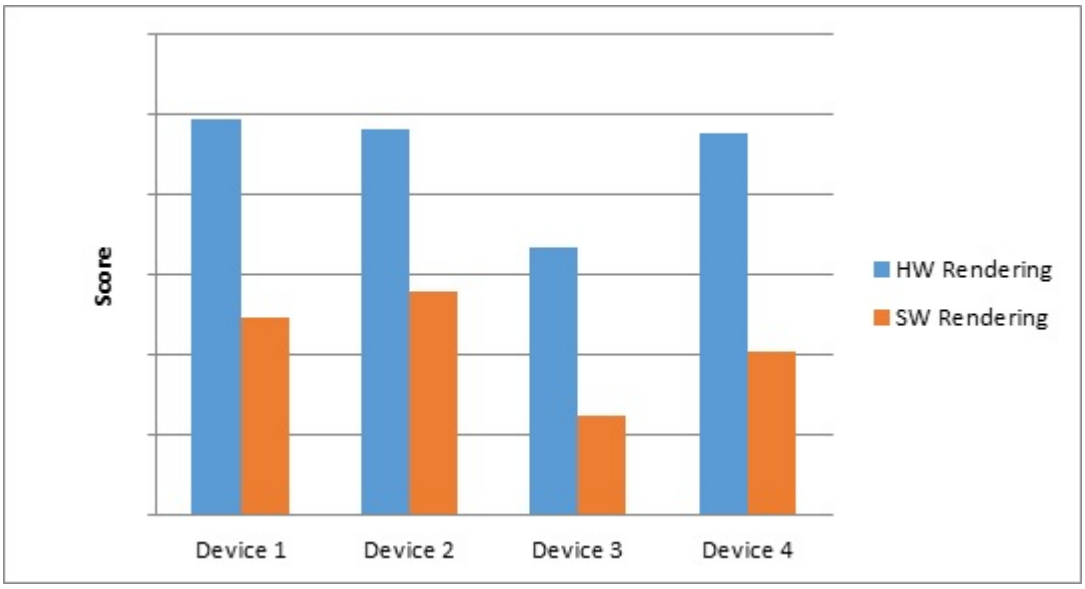

Figure 4.2: Quadrant 2D - Score Comparison

b. **HW Accelerated Rendering/Force GPU Rendering:** In this case, the 2D operations are done on GPU instead of CPU. 2D operations are translated into equivalent 3D operations and process it in GPU. The performance will be faster and will reach 60 FPS [score of 1000]. But the GPU power will be more compared to SW rendering i.e. V_{NN} is more.

Considering all these a test was carried out on all the platforms of Intel to see performance/ Watt for HW Rendering and SW rendering. Here, performance is nothing but the score which is reported at the end of the test. This score is calculated from the FPS of the test running on the platform.

Figure 4.2 compares score for HW and SW rendering for different Intels android devices. It can be seen that HW rendering scores are better than SW Rendering.
For a portable device it is not only performance that matters but, better performance at lower power consumption is required. For considering power and performance at a time, unit of performance/watt (perf/W) is used. Goal is to achieve perf/W as high as possible.

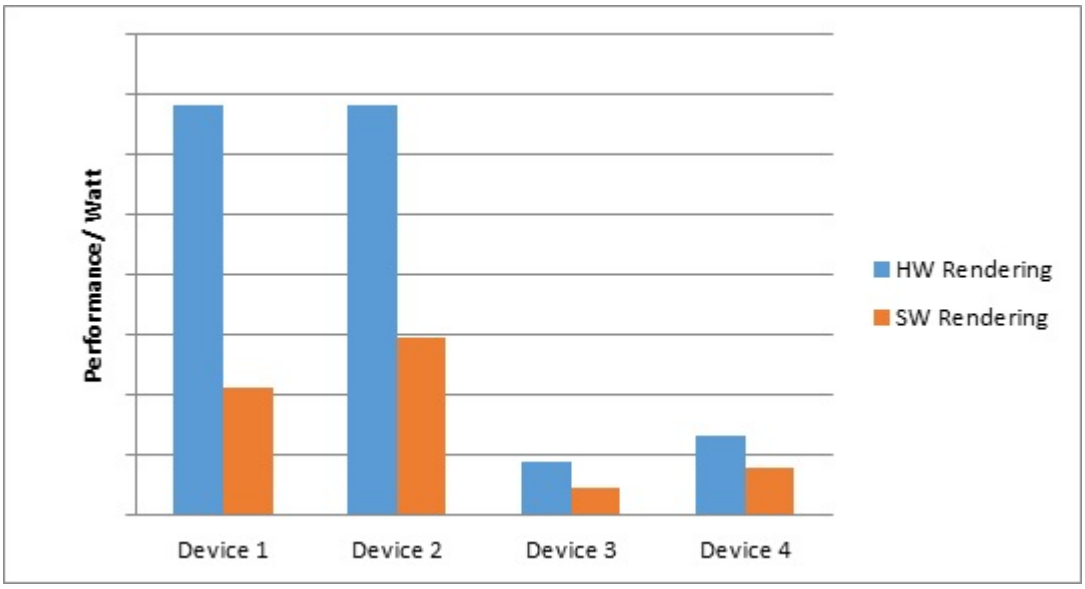

Figure 4.3: Quadrant 2D - Performance/Watt Comparison

Figure. 4.3 shows perf/W for different Intels device with SW and HW rendering. Only platform power is taken into consideration as it includes CPU as well as GPU power.

4.4.1 Conclusion

HW rendering gives better performance/Watt. This is because, due to hardware rendering performance increase and at the same time load on CPU decrease so power consumption on CPU decrease and power on GPU increase. As a result power remains almost constant and performance increase. So, performance/Watt improves.

4.5 Case Study: Graphics Benchmark - Characterization with Display Resolution

This case study presents characterization with display resolution. Performance changes with display resolution because, as number of pixel increases; GPU has to do more task and so, performance decrease with increasing resolution. Also, GPU task increase so power increase. This case study attempts to understand and answer following is-

sues.

a. Correct/Best Display Resolution

b. Behaviour of Graphics Benchmark with Display Resolution

c. Scaling Ratio with respect to Perforamance and Power

There are benchmarks which scales with display resolution but few benchmarks shows no impact of resolution on performance or power as they are designed to render at particular resolution and then scaled to display resolution so, energy remains same. This case study presents characterization for both type of graphics benchmark. This analysis helps setting display resolution for next platform and helps predicting values for other display resolutions.

Also, characterization becomes important when performance is good i.e. 60 FPS for Graphics Benchmark. With 60 FPS, GPU must be getting some time to sleep so characterizing the sleeping time of GPU and how it affects the power is important. If two platforms with performance for some benchmark are 60 FPS, then it is the processing time that matters. How fast it completes the process and goes to sleep to save power becomes important . So, energy characterization becomes important.

Instead of the real workloads, benchmarks are taken for experiment as they are repeatable and it becomes easy to compare platform. Here, characterization is done on Intels platform.

4.5.1 Quadrant

Quadrant is a benchmark for mobile devices, capable of measuring CPU, memory, I/O, 2D and 3D graphics performance. Quadrant runs several tests to evaluate performance of different components of the platform. Graphics point of view 2D and 3D are important. Only 3D test is consider and presented here as they are the most computing task and it is possible to redirect the computing to CPU or GPU. To evaluate

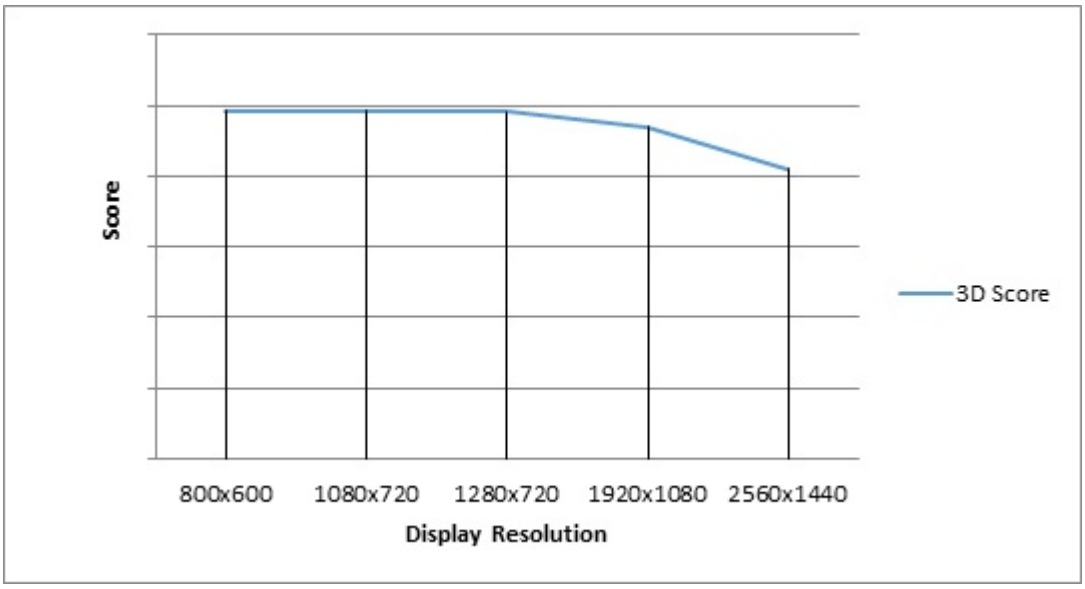

Figure 4.4: Quadrant 3D Score v/s Display Resolution

3D performance, Quadrant runs 3 test namely Corridor, Planet and DNA.The score for 3D for different display resolution is as shown in Figure. 4.4.

From the figure, it can seen that 3D score scales with resolution. As resolution increases performance drops. But out of the three tests running which test is/are the reason for drop and what is the drop in FPS is important. Characterization of FPS is carried out as shown in Figure. 4.5.

Figure. 4.5 shows FPS for DNA is same and maximum for all the cases. Till 1080 pixel resolution even Planet test is also good. But there is a large drop of FPS for Corridor at 2560 x1440 resolution. To under the reason of such behaviour, Corridor 3D test is analysed in details. First Graphics C State residency is analysed and results are as shown in Figure. 4.6

From Figure. 4.6, even with maximum resolution, GPU has some time to sleep. So, GPU is not the bottle neck here. To understand the behaviour further, memory BW for all the 3 tests is carried out and Figure. 4.7 shows the results.

In Figure. 4.7, all tests shows same behaviour but for corridor at 1080 pixel, it reaches the peak and then if resolution increases; BW become bottle neck and performance drop and so the BW. Even for Planet; it is the same case but still it is less than

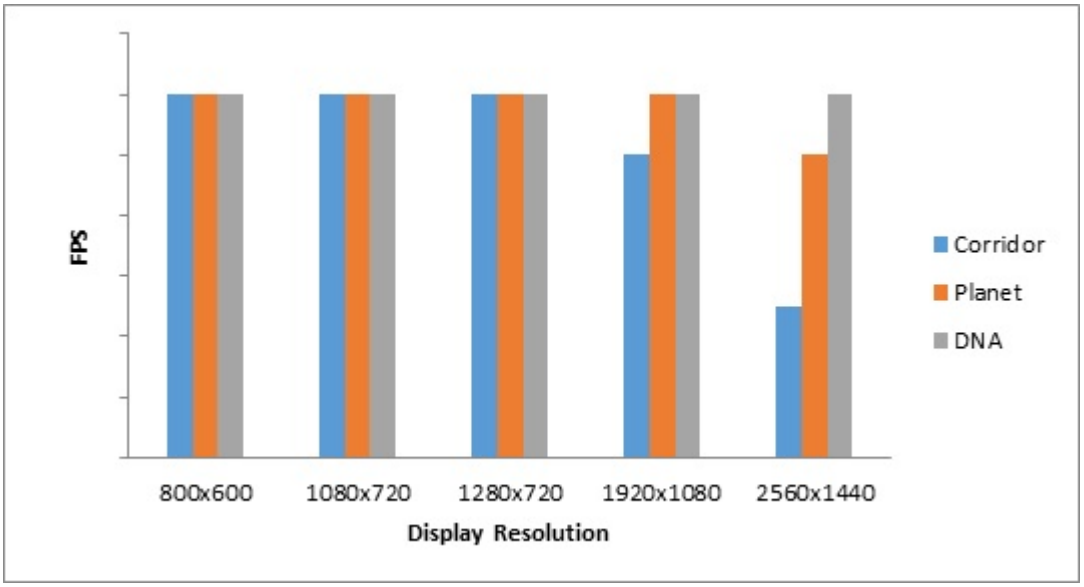

Figure 4.5: Quadrant 3D Score v/s Display Resolution

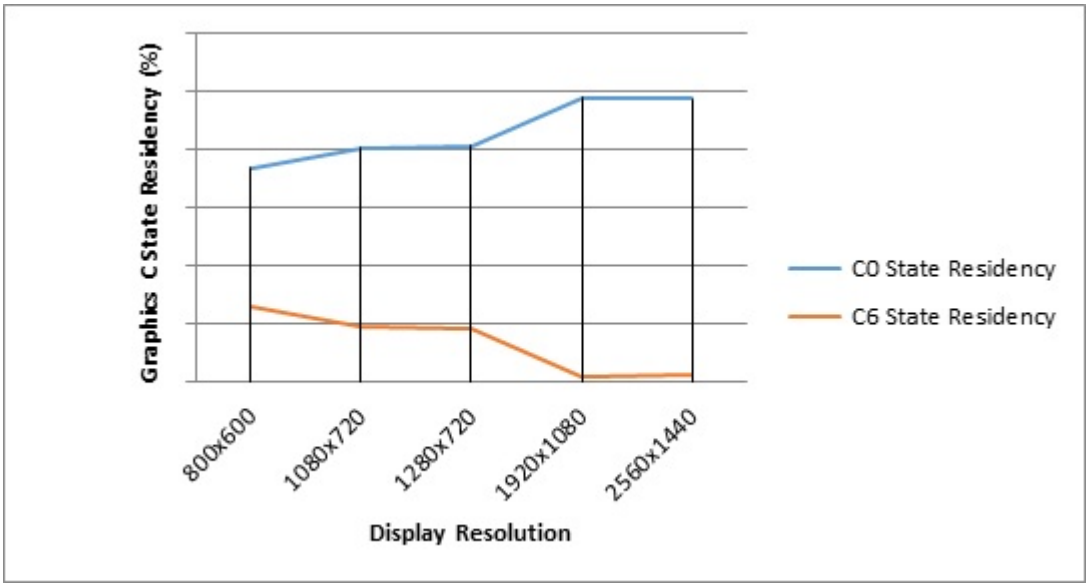

Figure 4.6: Quadrant 3D - Corridor Graphics C-State Residency

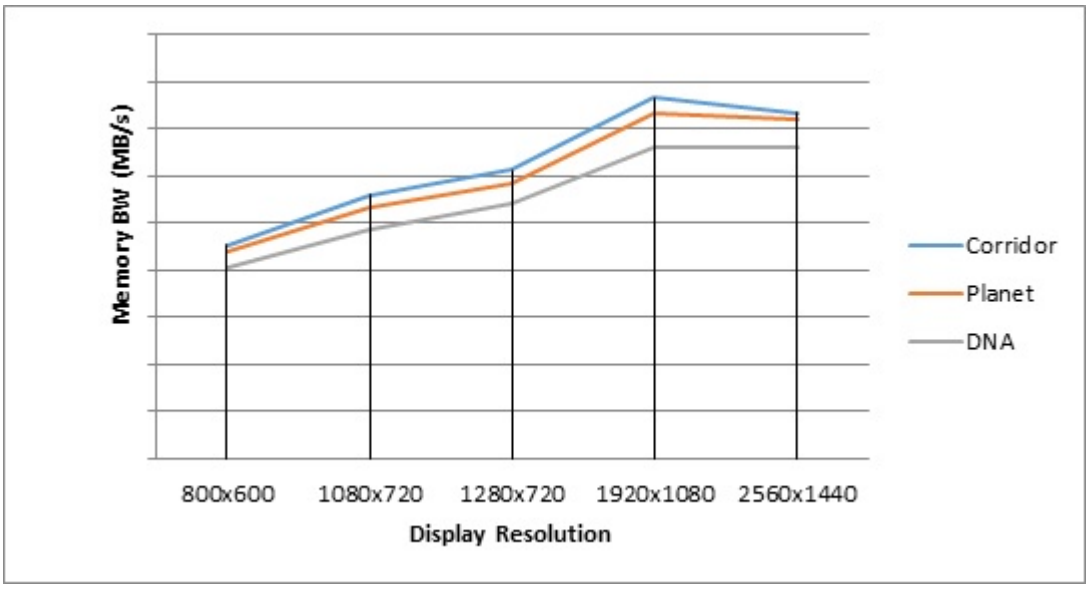

Figure 4.7: Quadrant 3D - Memory BW analysis

maximum available. For DNA, BW is comparatively less and so does not affect performance.

4.5.2 AnTuTu

AnTuTu is a Graphics Benchmark same as Quadrant but here 2D and 3D by default goes through HW rendering. For testing, 2D AnTuTu have 2 tests; Particle and Spirit whereas for 3D it runs only one test, Orge Activity. Figure. 4.8 shows 2D and 3D score with Display Resolution. 2D score is in primary axis in terms of hundreds and 3D score is in secondary axis in terms of thousands.

The 2D test runs multiple tests. So, to characterize the performance for individual test becomes important. Figure. 4.9 shows the FPS for all the individual test for 2D and 3D.

From Figure. 4.9, it is clear that 2D particle test is the main test due to which overall score is less. To understand the behaviour of Particle test Graphics C_0 residency is measured, which is as shown in Figure. 4.10

For Particle test even with lowest resolution Graphics is saturated. So, clearly Graph-

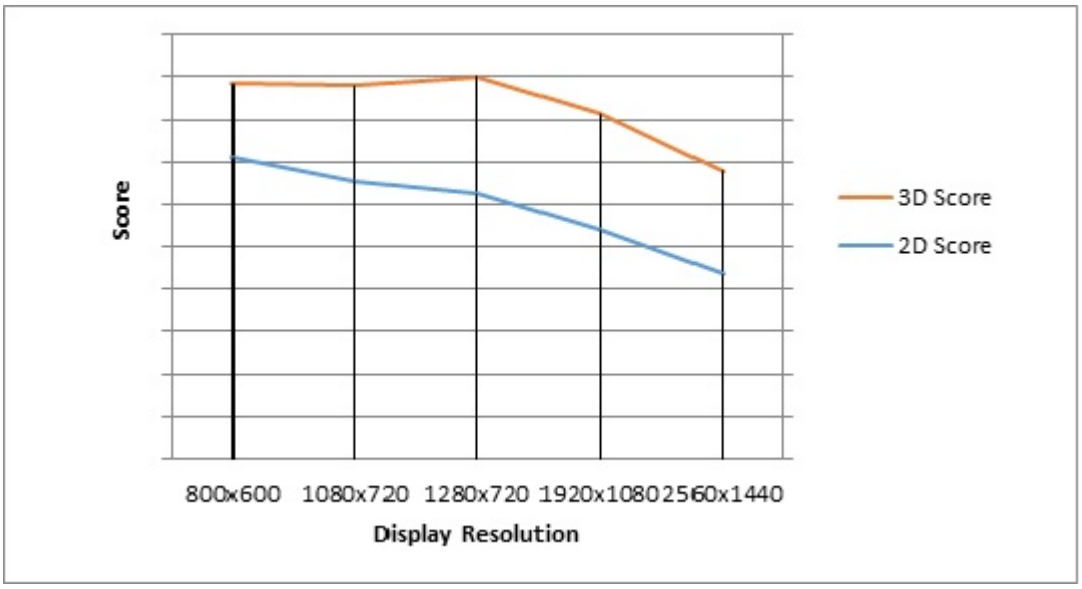

Figure 4.8: AnTuTu - Score v/s Display Resolution

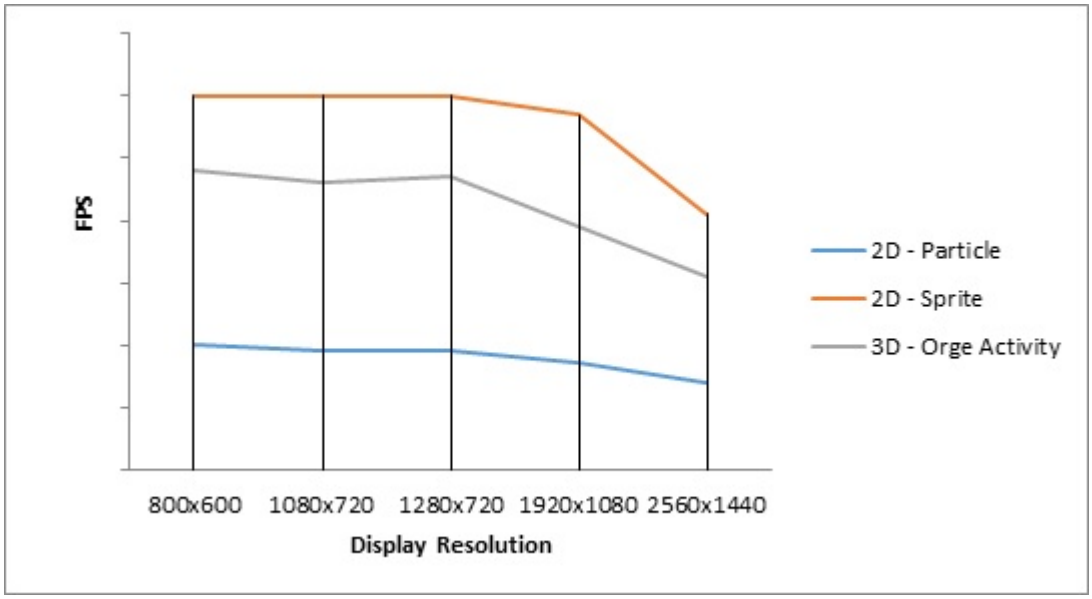

Figure 4.9: AnTuTu - 2D and 3D test FPS v/s Display Resolution

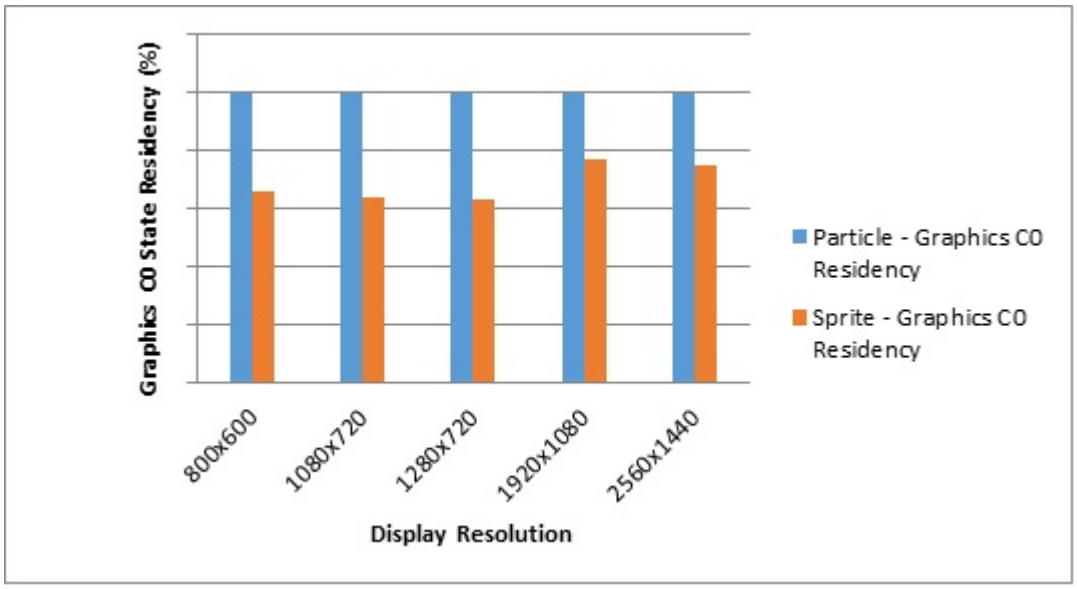

Figure 4.10: AnTuTu 2D - Graphics C_0 Residency

ics is the bottle neck.

4.5.3 Basemark X

Basemark X workloads is a Graphics benchmark designed according to the usage profile of modern and future 3D games targeted for smart phones and tablets. Basemark X features game like content, including particle effects, advanced lighting effects and post processing. It has two rendering options, Online and Offline. Figure. 4.11 shows the FPS for Online as well as Offline rendering.

Offline rendering does not show any performance change with resolution as it is always rendered at particular resolution; here 1080 pixel and then scales to the display resolution. But for online rendering this is not the case, FPS drop is almost linear with the increasing resolution. To under such behaviour of Onscreen rendering Figure. 4.12 shows the Memory BW characterization.

Memory BW for offscreen is almost constant but for onscreen it continuously increasing with initial BW itself is high, which becomes the bottle neck for lower performance.

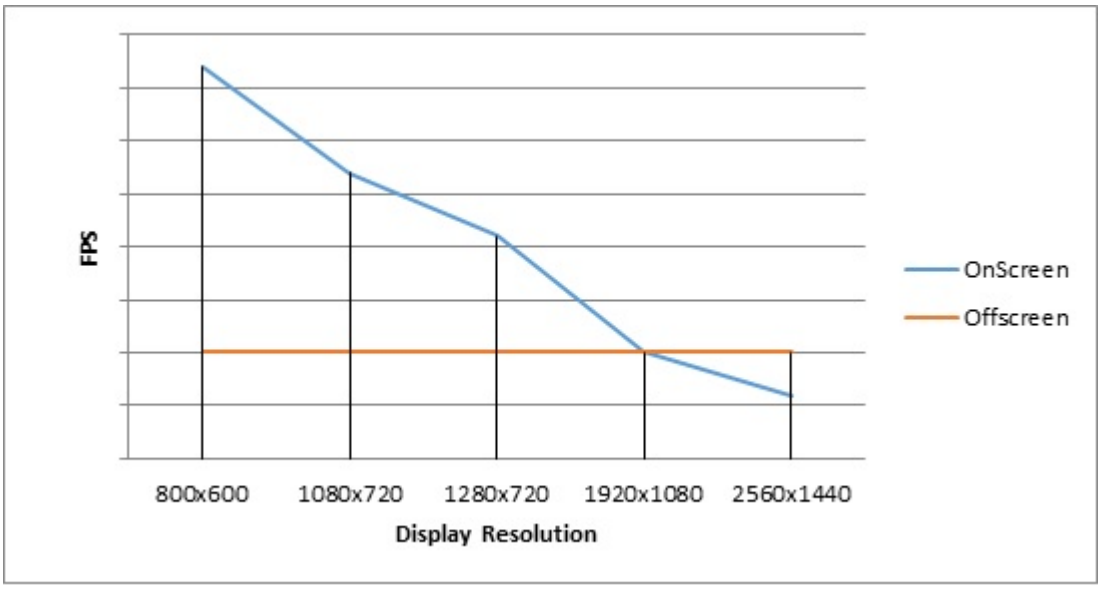

Figure 4.11: Basemark X - FPS v/s Display Resolution

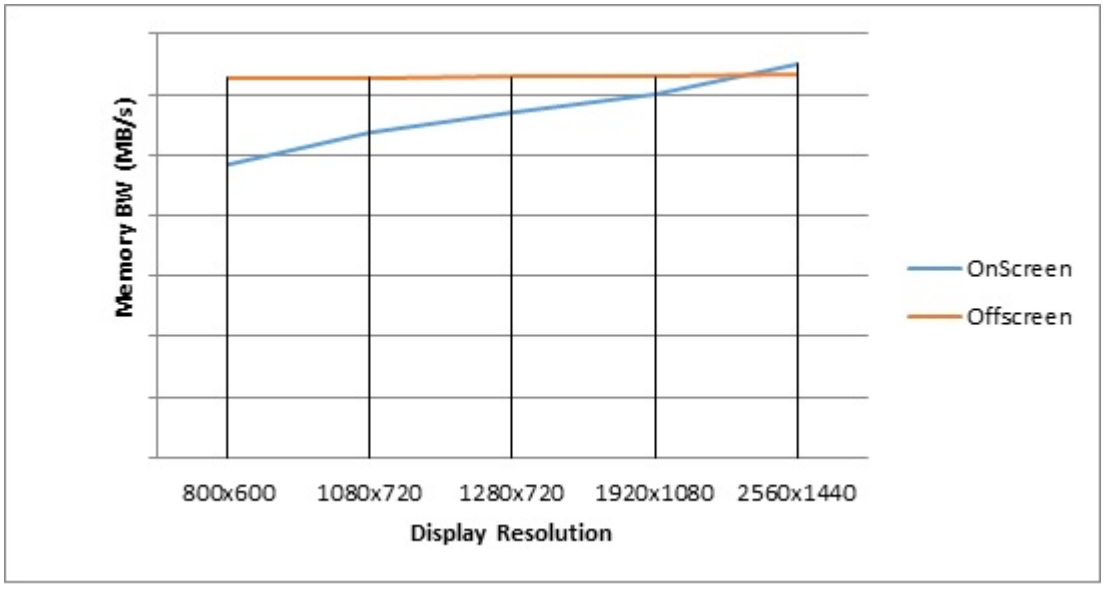

Figure 4.12: Basemark X - Memory BW Characterization

4.5.4 Summary and Conclusion

Graphics benchmarks show a huge performance drop at higher resolution. As display resolution increases performance drop is also increases. Figure. 4.13 summaries all the benchmark discussed above showing performance drop for increasing display pixels from x to 8x.

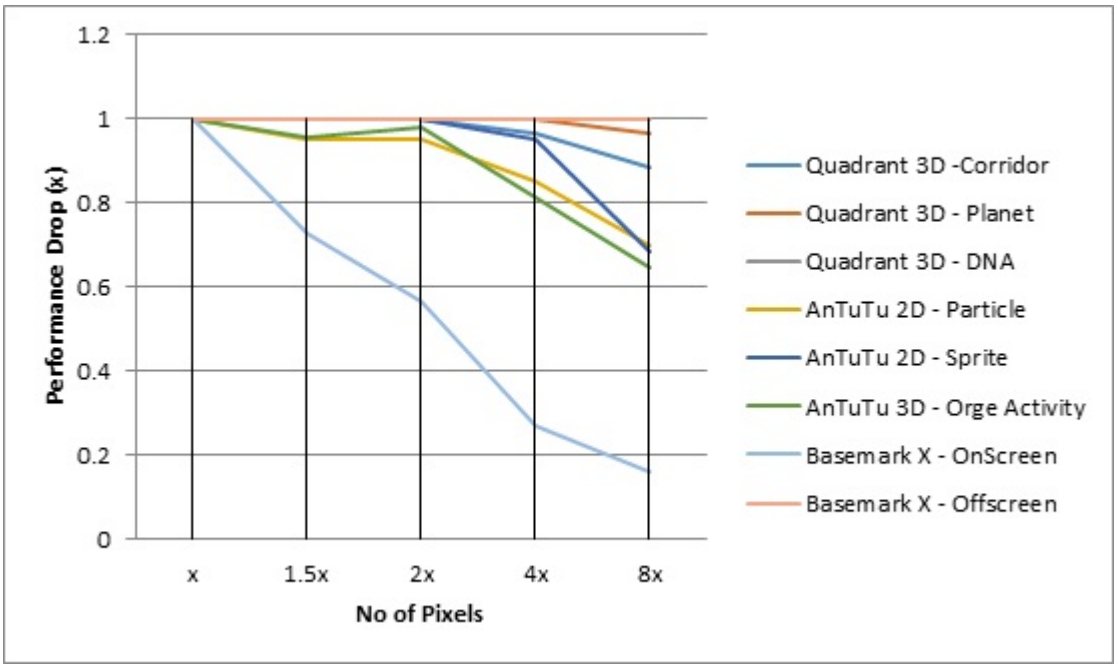

Figure 4.13: Performance Drop Characterizations with Increasing Display Pixels

Figure. 4.13 shows that till 2x display pixel values, performance drop is not significant except for Basemark X online but for 4x and 8x performance drops significantly. This helps to characterize performance for even higher resolution and helps to set proper display resolution for next platform. This case study suggests 1080 pixel display resolution for next generation platform which gives better performance and display to user at acceptable power.

Chapter 5

Platform Architecture Modeling & Simulation

Modelling and simulation (M&S) is getting information about how something will behave without actually testing it in real life. More generally, M&S is using models, including emulators, prototypes, and simulators to develop data as a basis for making technical decisions. M&S helps to reduce cost and increase the quality of products and systems.

Using simulations is generally cheaper and safer than conducting experiments with a prototype of the final product. Simulations can be more realistic, as they allow the free configuration of environment parameters found in the operational application field of the final product. Simulations allow setting up a coherent synthetic environment that allows for integration of simulated systems in the early analysis phase via mixed virtual systems with first prototypical components to a virtual test environment for the final system.

Modelling of actual mobile platform is done and then it is simulated for a particular use case for power and performance analysis. This help to have a picture of energy for the next platform without having the actual hardware. But it is necessary to check the stability and accuracy of the data from the simulation. This chapter presents a

SYSTEM approach for modelling platform and a "Real Life Use-Case using a Platform Architect Tool. It attempts to present a model for a platform whose silicon is available and a real life use case like Video Playback using SYSTEM Task Flow approach. It then shows the correlation of simulation results with measured results for accuracy of the approach.
Section 5.1 is about the System Task Flow approach and its benefits. Section 5.2 describes on PA Tool, how it supports the Task Flow approach, its features and output. Section 5.3 and 5.4 includes Platform modelling and Workload modelling. Section 5.5 presents the simulation results and correlation with measured.

5.1 SYSTEM Task Flow Approach

Platforms are getting increasingly complex due to the advances of technology. As discussed in section 2.3, platform is an integrated set of ingredients such as the processor, the chipset, a lightweight form-factor for mobility, technologies to provide great battery life, enabling the ability to work anywhere you want without needing a power socket, wireless capabilities that enable the ability to connect in almost any location equipped with private and public hotspots, and sophisticated applications that use the mobile form-factor efficiently. The final shipped product is the sum of a set of parts that provides the valuable end user experience - it is not just a device, but a platform [2]. Evaluation of whole system/platform performance and power is a critical part of system optimization and validation. The two main performance metrics are DDR transaction or throughput. Throughput can be expressed as MB/s. Power can be estimated using residency of each IP in the highest power state.

There are various approaches for modelling and simulation of workload, static and dynamic. Static is simulating without the actual use-case like a high level spread sheet analysis and dynamic includes simulation of actual workload. But there remains a significant gap to analyse real world Use Cases at SYSTEM level and appropriately

co-design HW/SW for optimal Power/Performance. It is important to analyse and optimize for real world Use cases accounting for end-to-end SYSTEM behaviour using both measurements and modelling approaches. This approach is a static approach where the details of various parameters of particular workload is used but has advantages as compared to static approach. In this chapter a workload simulation is presented which shows that a real life use-case like Video Playback can be modelled on a particular system. It allows us to examine the performance behaviour of very complex use cases before the hardware prototypes are available. This is essential when feasibility of architecture needs to be evaluated or to optimize the system parameters for some application scenario.

In Task Flow approach each workload is divided into several system level logical tasks. These tasks have specific bandwidth and processing time as per the platform. It may or may not scale with platform depending on the platform specification changes. They are interconnected to form a logical flow. These tasks are interdependent running serial or parallel according to the architecture of the platform. Also, each task may be a processing task or a memory task; mapped to different hardware in the system. All these information is given as an input in this approach. Few dummy tasks are included in the system for replicating a real use case scenario. User interaction or program interrupts are taken care of and is modelled using dummy tasks. A generic flow of Task Flow approach is as shown in Figure. 5.1 . With this approach it is very easy to add features to a workload like HDMI Display for Video Playback, and increase the sensitivity to various parameters of the system and workload. Detail explanation of this with example is discussed in section 5.4.

This System level Task Flow approach has many benefits. It represents the behaviour of the real applications. It is possible to give priority to the task as in real use-case which is not possible in high level spread sheet analysis. This helps to study the peaks in the system. Parallel execution of task possible and this will help to study the performance improvement. It is also possible to add entry and exit latency

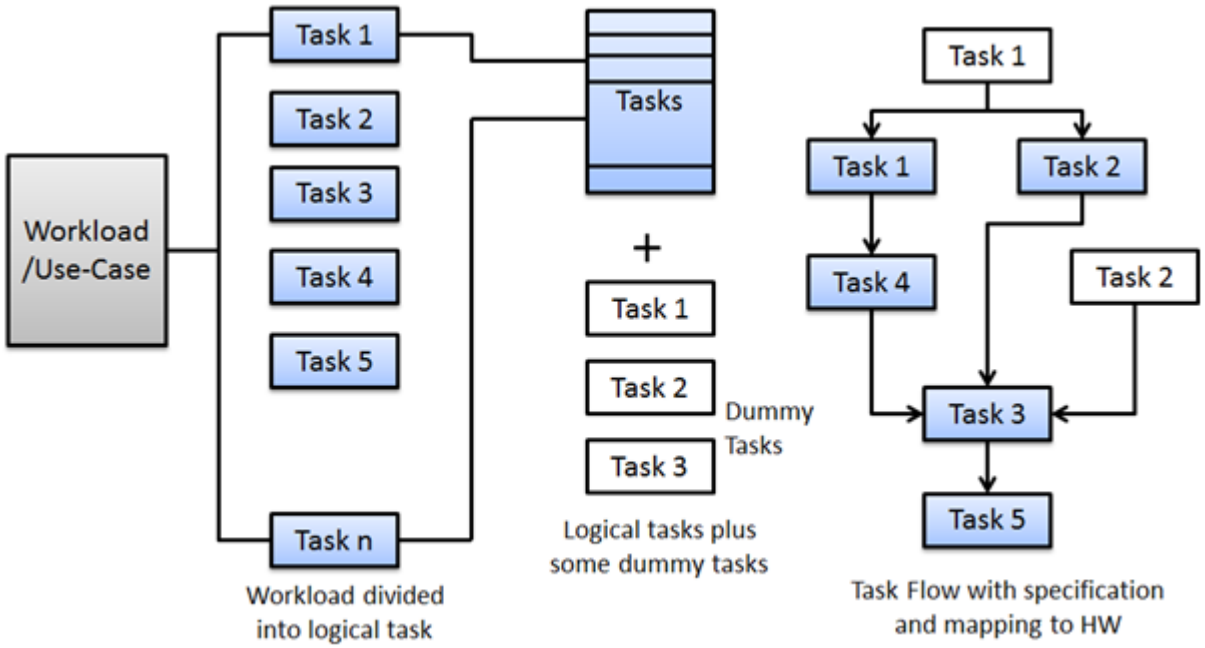

Figure 5.1: Generic Workload Task Flow

for each CPU state and so power analysis is also possible. Task flow approach is a generic approach where adding a feature like CPU C states, GFX C States, GFX frequency, CPU frequency, etc. is feasible. This approach is very useful to assess system scalability of both HW and SW, as large number of derivative SOCs and Platforms are produces to meet specific customer needs. Task Flow models helps to establish a common language between SOC, Platform Architects, SW developers and Validation Engineers, and enable significant improvement in platform definition and HW/SW development. This approach is a significant step in studying Use Cases at a much higher abstraction than cycle accurate models and thus enables to take Systems approach for Power/Performance efficiency - an absolute need of the hour to win Phone/Tablet segments.

Platform Architect tool from Synopsys supports the system approach, is designed according to Y-chart model [11, 12 and 13] and consists of the following components (Figure. 5.2):

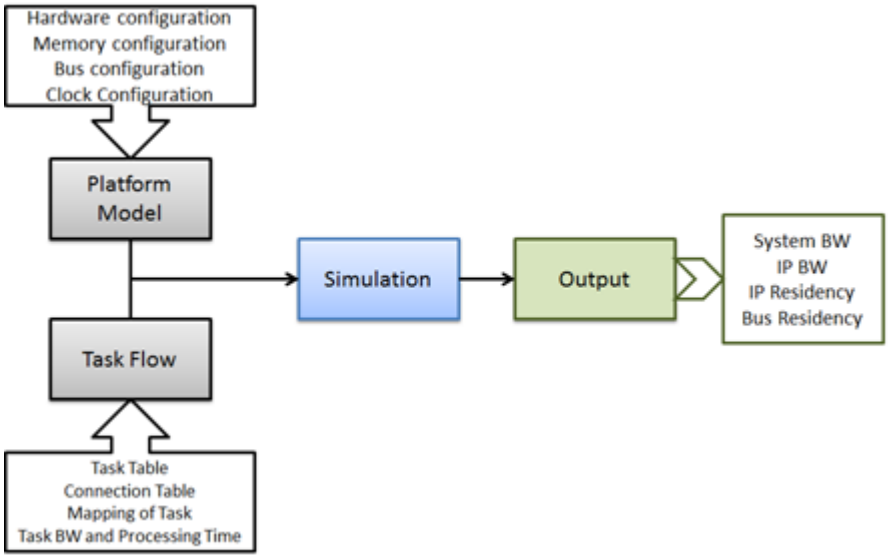

Figure 5.2: Y-Chart Model of PA Tool

a. HW (Hardware) Topology

b. Task Flow

HW topology is Platform Architecture specific; mainly concentrate on modelling the actual platform parameters and configuration whereas Task flow is of particular Use-Case like Video Playback, Audio Playback, Image Capture, etc.

5.2 Platform Architect Tool

Platform Architect tool is developed by Synopsys and is designed according to Y-chart model. It has one input as platform model and other input as workload model. Also, it has inputs for CPU scheduling algorithm for multi-core processors. It supports task flow approach as describe in section 5.1. For a user point of view as described it basically require two inputs namely, HW model and Task flow. CPU scheduling algorithm can be changed but at present the default algorithm as provided by the Synopsis is used.

```
set ::vpu_names          {   CPU0    CPU1    GFX     SCHEDULER   }
set ::vpu_buswidth       {                                       }
set ::vpu_freq           {                                       }
set ::vpu_bus            {                                       }
set ::vpu_prio           {                                       }
set ::vpu_gpu_cores      {                                       }
set ::vpu_burst          {                                       }
set ::vpu_read_credits   {                                       }
set ::vpu_write_credits  {                                       }
set ::vpu_dma            {                                       }
set ::vpu_vc             {                                       }
set ::vpu_categories     {                                       }
set ::vpu_power_off      {                                       }
set ::vpu_power_idle     {                                       }
set ::vpu_power_active   {                                       }
set ::vpu_enable         {                                       }
```

Figure 5.3: Hardware Configuration

5.2.1 Hardware Model

Hardware model file consist of several information about the actual hardware model which is to be built. It is basically in 4 sections.

a. Hardware configuration

b. Bus configuration

c. Memory configuration

d. Clock configuration

5.2.1.1 Hardware Configuration

For any simulation a hardware model or the environment on which it will run is needed. Hardware configuration section in the model file contains of information regarding different fabrics, IPs connected to fabric, fabric frequency, fabric bandwidth, memory configuration, etc. A sample Hardware configuration is as shown in Figure. 5.3.

In this configuration there are 4 IPs / VPUs (Virtual Processing Unit), CPU0,

CPU1, GFX and SCHEDULER Parameters

- vpu_buswidth : Number of data bits that connect the VPU to the bus
- vpu_freq : The frequency in MHz that the VPU clock is running
- vpu_bus : To which bus/fabric this VPU is connected to
- vpu_prio : The priority used by the bus to order different transactions from different VPUs (0 highest)
- vpu_gpu_cores: Number of cores each VPU have. Not used in this lab so, 0
- vpu_burst : Data in byes that the VPU can send in a single transaction
- vpu_*credits : The VPU read and write bus credits. This defines the number of request the VPU can put before any request is served. A VPU here a send 15 read and write request before waiting for 1st one to be served.
- vpu_dma : Not used in this lab use 0
- vpu_vc : VPU virtual channels, prioritize virtual channels list, you can map memory transaction to any channel of your choosing
- vpu_categories: Depends on vpu_gpu_cores. GPU for multi core and CPU for single core
- vpu_power_* : Power in W for the different power states
- vpu_enable : Must be 1

5.2.1.2 Bus Configuration

This defines the bus/fabric used in the model. It also defines the fabric connection, frequency and bandwidth.A sample Bus configuration is as shown in Figure. 5.4.

```
set ::bus_names          {   MAIN    PSF }
set ::bus_freq           {                }
set ::bus_buswidth       {                }
set ::bus_prio           {                }
array set ::bus_connection {
                PSF MAIN
}
```

Figure 5.4: Bus Configuration

In this configuration there are 2 buses, MAIN and PSF

Parameters

a. bus_freq: The frequency in MHz that the bus clock is running

b. bus_buswidth: Number of data bits that connect the bus to the target bus (if exists)

c. bus_prio: The bus priority used by the target bus to order different transactions from different VPUs/buses (0 highest)

d. bus_connection: This array describe the bus topology, in this case (PSF MAIN), the PSF is the initiator and the target is MAIN, any number of pairs can be added to describe the topology

5.2.1.3 Memory Configuration

This section defines the memory. It defines the type of memory, frequency, bandwidth, etc. A sample memory configuration is as shown in Figure. 5.5.
In this configuration only one memory, SRAM is used.

Parameters

- bus_buswidth : Number of data bits that connect the memory to the bus
- bus_freq : The frequency in MHz that the memory clock is running

```
set ::mem_names          {   SRAM        }
set ::mem_buswidth       {               }
set ::mem_freq           {               }
set ::mem_prio           {               }
set ::mem_bus            {               }
set ::mem_type           {               }
set ::mem_addr_width     {               }
set ::mem_power_idle     {               }
set ::mem_power_read     {               }
set ::mem_power_write    {               }
```

Figure 5.5: Memory Configuration

```
set ::clock_freq    { 800 }
```

Figure 5.6: Clock Configuration

- mem_bus : To which bus this memory is connected to
- mem_type : The memory type, currently supported SRAM and DDR only
- mem_addr_width: Address width in bits of this memory
- mem_power_* : Power in Watt for the different power states

5.2.1.4 Clock Configuration

This section defines all the clocks used in the system. All frequency is given in MHz.

5.2.2 Task Flow

Task Flow is specific to workload and has two files as an input to PA tool.

a. Task Graph Configuration

b. Mapping file

These are .csv files each defining various parameters.

5.2.2.1 Task Graph Configuration

Task Graph is the main input to the tool, since after building the platform, the use-case desired to run is defined in a task graph. Task Graph file including 4 tables:

- Task Table: Used to define all the tasks and general parameters on each one of the tasks such as wait latency, number of instances and priorities
- Connection Table: Used to define the connectivity between the tasks (which task is enabler to which task)
- Function Table: Used to define the behavioural model of the task. Whether its a memory transaction task or a processing task
- Memory table: Used to define all memory transactions address widths

Task Table

Task table defines different task used for a particular workload. Any name can be given to a task. A sample task table is as shown in Figure. 5.7.

# task table								
task_name	instances	start_delay_in_ns	wait_delay_in_ns	iterations	auto_stop	job_id	priority	or_gating
start		0	0	2				
read		0	0					
work		0	0					
write		0	0					
end								

Figure 5.7: Task Table

Parameters Description (Table I)

Connection Table

Connection table specifies the whole data flow for the task. How the different tasks are connected and who will initiate data for whom. A sample connection table is as in Figure. 5.8.

Name	Data Type	Description
instances	Iinteger	Specifies the amount of task instances this task block will span
start_delay_in_ns	Integer	Specifies the initial delay and is executed when the task start for the first time
wait_delay_in_ns	Integer	Specifies the self-activation delay and is executed after each iteration
iterations	Integer	Specifies the number of activations during a simulation
job_id	Integer	Specifies the job ID to which task instance 0 of this block belongs to. This enables multiple task instances to be associated with one job for simultaneous start and stop
priority	Integer	This parameter is used to give priority to the task running on same IP/VPU. It influences the priority scheduler
or_gating	Boolean	If false (default), the task executes when all the get ports have data. If true, the task executes as soon as one get port has data

Table I: Task Table - Parameter Description

#connection table

task_name	destination	put_port_ name	get_port_ name	p_put.rate_ samples	p_put.rate_ activations	p_put.offset	p_get.rate_ samples	p_get.rate_ activations	p_put.init_put_ samples	size	p_get.get_ samples	auto_resize
start	read											true
read	work											true
work	write											true
write	end											true

Figure 5.8: Connection Table

Name	Data Type	Description
rate_samples	Integer	It determines the number of samples generated per activation
p_put.rate_activations	Integer	One source token activates p_put number of destination token
p_get.rate_activations	Integer	After 4 source tokens, a single destination token is activated
init_put_samples	Integer	Specifies the number of samples generated initially
offset	Integer	This allows to shift the activation in which the put samples are generated
p_get.get_samples	Integer	Specifies number of samples to wait before finish
aut_resize / size	Boolean/ Integer	Used to define the buffer between two tasks Task_A pushes Task_B and it sends 10 tokens to it, if Task_B is stuck for some reason, Task_A can either get stuck after N iterations (N $<$ 10) or continue running and pass the 10 tokens. If true then Task_A never gets stuck and it continues to run. If Auto resize is false then the size comes in the picture and defines the N iterations Task_A can pass before it also gets stuck.

Table II: Connection Table - Parameter Description

Parameter Description (Table II)

Function Table

Function table specifies the details of each task. It specifies the type of task and processing cycle for processing task and number of bytes for read and write task. A sample function table is as shown in Figure. 5.9.

#function table

task_name	function_name	function_type	processing _cycles	fetch _ratio	store _ratio	load _ratio	memory _name	burst_size_ in_bytes	read	total_data_size_ in_bytes	data_size_per_ activation_in_bytes	call_port _name	instances
read	rd	scml_tm_mem_function					rd1	64	1	64	64		
work	proc	scml_tm_cpu_function	5										
write	wr	scml_tm_mem_function					wr1	64	0	64	64		

Figure 5.9: Function Table

Parameter Description

a. Function Name : rd,wr or proc , based on the functionality of the task

b. Function Type : for rd/wr tasks must be scml_tm_mem_fucntionand for proc task must be scml_tm_cpu_function

c. Processing Cycles : Number of cycles needed to process processing task

d. Memory Name : Used for rd/wr tasks and defines the symbolic naming of the memory transaction for that task

e. burst_size_in_bytes : Is responsible for quantity of memory calls

f. read : 1 for rd tasks 0 for wr tasks and "" for proc tasks

g. total_data_size_in_bytes : Is responsible for the amount of activation after which the accesses start at local address 0x0 again

h. data_size_per_activation_in_bytes : Specifies amount of bytes transferred per activation

Memory Table

This is used to define all memory transactions address widths. A sample table is as shown in Figure. 5.10.

memory_name	addr_width
rd1	10
wr1	10

Figure 5.10: Memory Table

5.2.2.2 Mapping file configuration

For any taskgraph to simulate on the given model, mapping file is needed. This defines which task will run on which IP/VPU. A sample mapping file is as shown in Figure. 5.11.

# task table		
task_name	task_id	mapping
start		
read		CPU0/CPU0_VC0_RD
work		CPU0/CPU0
write		CPU0/CPU0_VC0_WR
end		

Figure 5.11: Mapping File

5.2.3 Steps to Build, Simulate and Analyse

Once the model file and task graph is ready, next step is to build the platform and simulate it. Following are the steps.

a. Run PA

   $> pct &

b. Create platform

   Plugins -> vpp -> create system. This flow will take the information from

model file, platform_parameters.tcl and create the system. A sample platform created by tool is shown in Figure. 5.12.

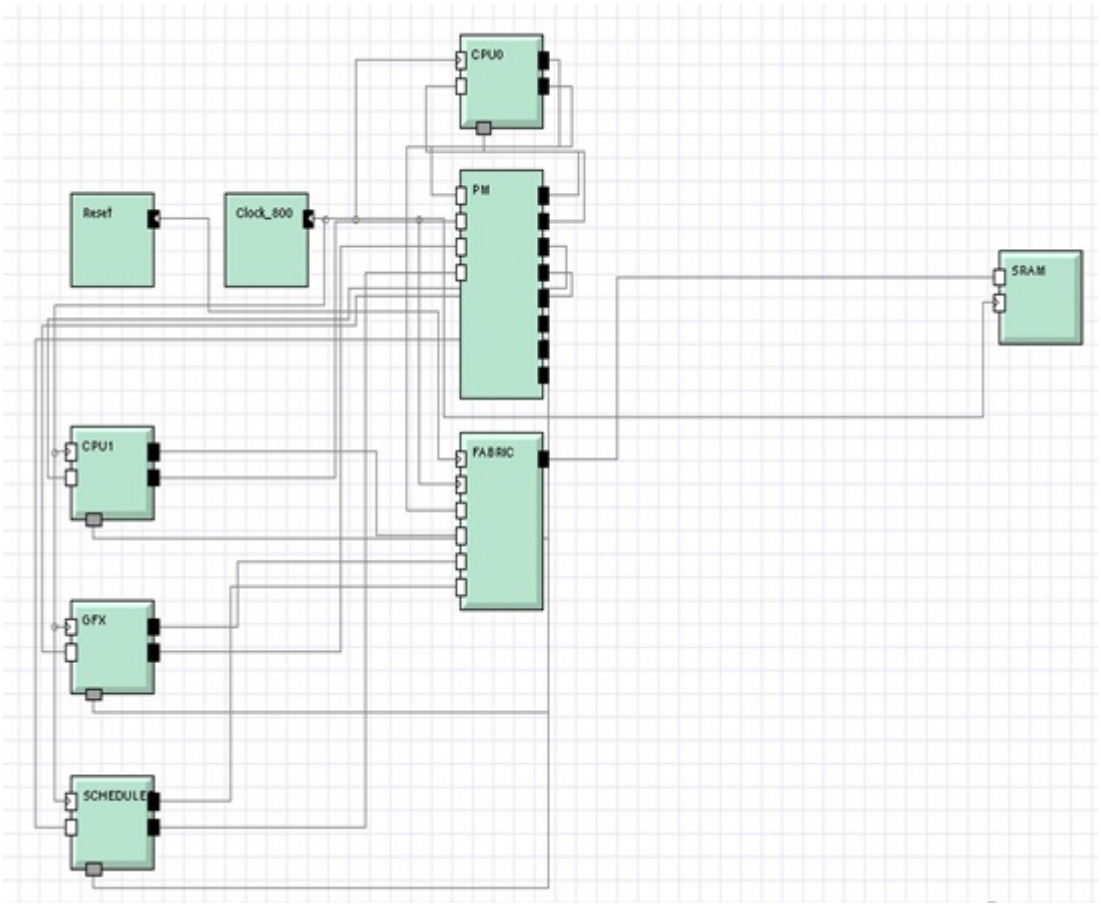

Figure 5.12: System Model

c. Create Task Graph

The Application parameters in the HW parameters tree should match the task graph which needs to be loaded.(Figure. 5.13)

Create the task graph by running: Plugins -> vpp -> create tg. This will create the task graph as defined in the connection table. A sample task graph is as shown in Figure. 5.14.

d. Build: Plugins -> vpp -> build

e. Simulate: Plugins -> vpp -> simulate

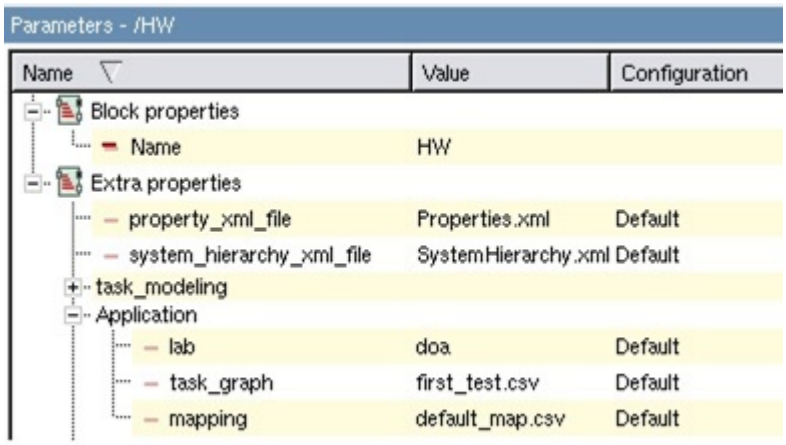

Figure 5.13: HW Parameters

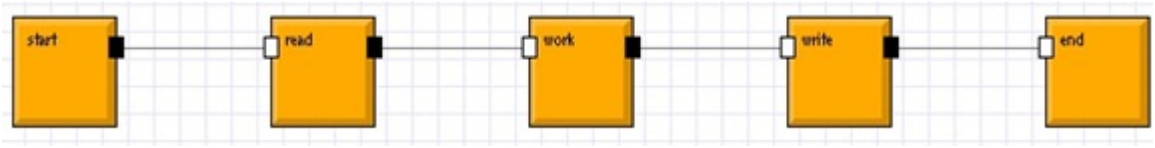

Figure 5.14: Task Graph

f. Analyze: Plugins -> vpp -> analyze

This will open VP explorer, loading MCO Trace will show the time based simulation of each task (Figure. 5.15).

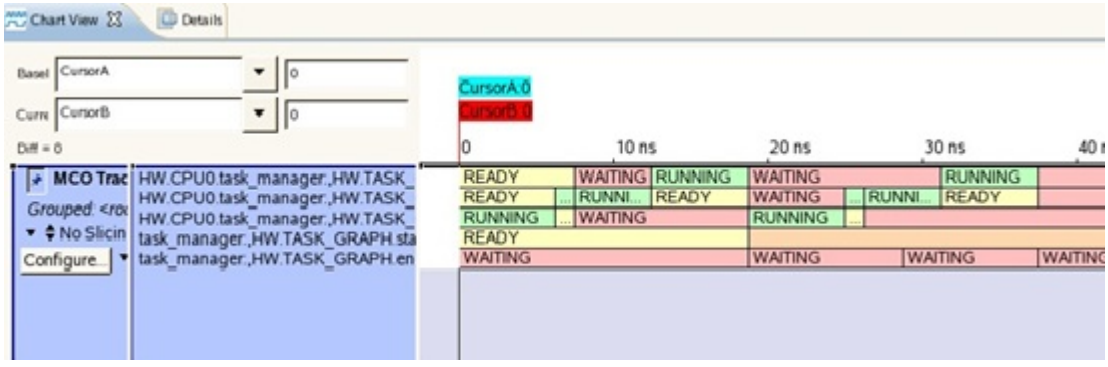

Figure 5.15: MCO Trace Sample Results

5.2.4 Features of PA Tool

- Syntax and number of parameters in any model file can be modified or added as per the requirement. Like as present there is no parameter for memory efficiency in the model file. So, simulation assumes that 100% memory is available. But in real scenario it is not the case. Particular use case can only use the 70% of the memory and then it starts degrading the performance. This can be handled by adding a parameter of memory efficiency in the model.

- Algorithm for scheduling of CPU task for multicore processor can be modified. At present it uses priority scheduling. Last core is given highest priority. Any thread is first mapped to last core. If other thread comes, it will be scheduled to next CPU core only if the last core is busy. This algorithm can be modified as per the requirement.

- It allows assigning priority to each task and to the IPs.

- Any task can initiate more than one token for destination and similarly after more than one token a single task can be initiated. This helps in batching or buffering. Like if in the video pipeline video decode takes maximum time, decode of frames can be done in advance as per the memory availablility and this improve the performance. This is done in real world use-case and PA Tool helps to model such scenarios.

- Any task specifications can be made sensitive to workload parameters and system parameters. For example, a task of Video Decoding can be made sensitive to change in clip resolution, FPS, etc. as real workload behaves.

- Tool presents the results for task as time based execution of the task. Parallel task or series execution as per the task flow can be easily viewed.

- It gives bus BW and IP BW in MB/s.

- It supports addition of post processing scripts for detail analysis as per requirement. Like each IP residency, total memory transactions in an interval, etc.
- It also supports to add power features like IPs connected to which rail and what is the voltage and frequency. This will help in power analysis.

5.3 Platform Modelling

Typically a platform is an integrated set of ingredients such as the processor, fabrics, IPs, memory, bus interconnecting different IPs and sophisticated applications that use the form-factor efficiently. As the emerging and the most hot spot today is the android based mobile/tablets, platform modelling is done for android based mobile platform. A generic mobile platform based on android operating system is as shown in Figure. 5.16.

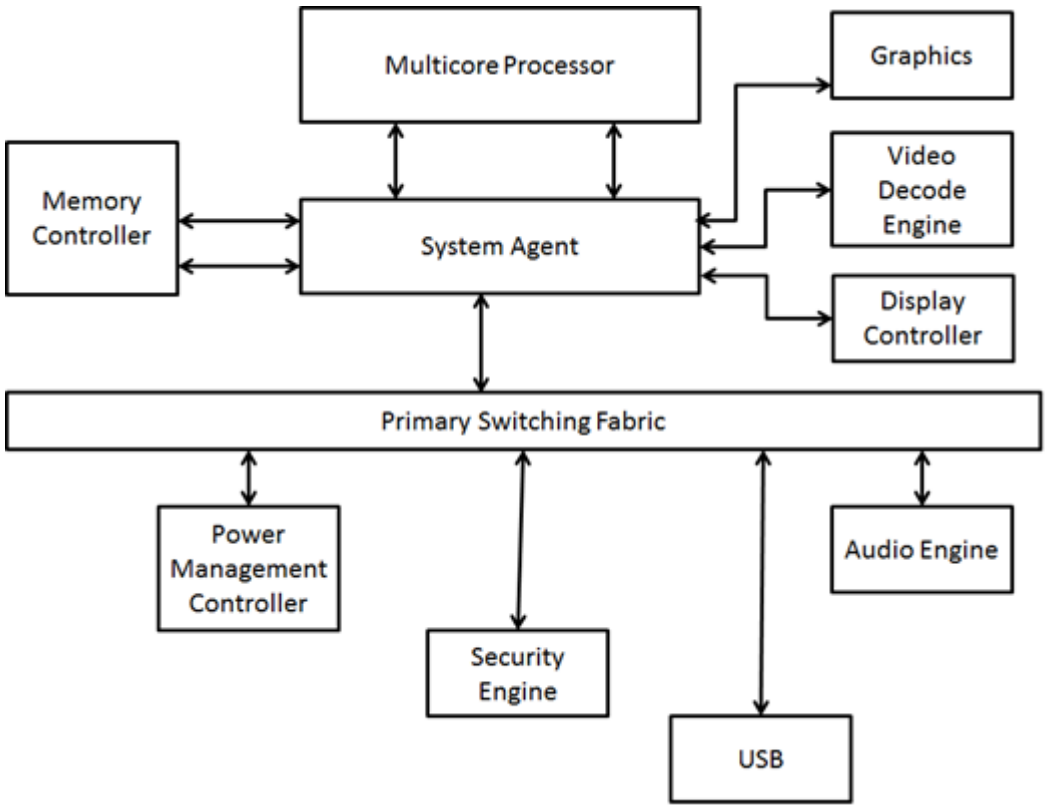

Figure 5.16: Generic SOC of Mobile/Tablet Platform [14]

At the centre of the "North Complex" above primary switching fabric is the system agent. The SA is the traffic cop that directs the flow of data between the major functional blocks in north complex. The system agent employs crossbar architecture, like a network switch, to ensure high-bandwidth communication from any one component to any other. The SA is linked to the processor cores using a point-to-point interface known as IDI. A multicore processor - main block of the SOC connected to system agent, consists of several processing units, interconnection, consists one or more memory interfaces and a group of peripherals. Often most resources are shared between processor cores but some of them may also be dedicated to a single unit. Simulation helps to analyse the performance of processor and helps to optimize it.
Also, as discussed the embedded systems and various computing devices such as smart phones, tablets, are often used for prolonged periods without being connected to a power outlet. This usage pattern poses a new set of challenges related to power and performance. The challenge is to balance the performance and power so that the device can be used for an acceptable duration within acceptable performance bounds. Performance without power considerations is meaningless, especially in the smartphone world. Thus a power analysis is equally important. Simulation helps to analyse the residency of each IP like CPU, GPU, Memory, Display, etc. in the different power states mostly C_0 and C_6.
As discussed in section 5.2.1, typically for modelling a platform, a topology diagram is required where it has information of different fabrics, IPs connected to fabric, fabric interconnection, IP frequency, IP priority, bus width and frequency, number of cores for each IP, virtual channels, priorities of virtual channel, memory configurations, clock configurations, etc. All this are given as an input for platform modelling. Basically for making the platform modelling simple only IPs used for a particular workload are modelled. For example, IPs required for Video Playback workload are CPU, GFX, Media, LPE, SDIO, G-unit, and Display and hence only these IPs and related fabrics and clock are modelled. But for a generic platform modelling supporting all workload, it is required to model every IP of the platform. Here for simplicity;

only IPs related to Video Playback are modelled.

5.4 Use-Case Task Flow Modelling

Workload or Use-case is the amount of processing that the computer needs to do in a given time. It is composed of particular application, operating system, drivers which in turn consist of data and control flow structure and their functions [15]. The objective of Use-Case modelling is to analyse the chosen use-case in terms of the task involved, functionality of the task, HW/SW execution and create a model that represents it as closely as possible to the real life. Modelling use-case correctly is important as it affects the result accuracy.

Here approach is to break the use-case into several tasks, model the connection, control and data flow for each. It also consists of mapping of the task to different hardware units of the platform. It models the actual flow structure with dependencies on system and workload parameters and hence the function of workload is dynamic. A generic Task Flow model of Video Playback on mobile devices is as shown in Figure. 5.17.

The whole video playback is divided into general task like eMMC driver loading, de-muxing (separating audio and video stream), audio and video codec (remove overheads), loading video drivers (if video decoding is done in separate hardware), audio and video decode, sync signal, frame flip, display driver and display. The specification for each task in terms of bandwidth and processing time is through measurement and from Architects. For simulation purpose few dummy tasks are included. Token Generation, Buffer Empty, End of Video Decode and Video Codec Request are mainly included for buffering the video frames. Start UI path is for profiling the user iteration while playing video which is player coming up when screen is touched.

The task flow and mapping also depends on Architecture of platform. For some architecture video decode is done by CPU whereas for some there is a separate HW for

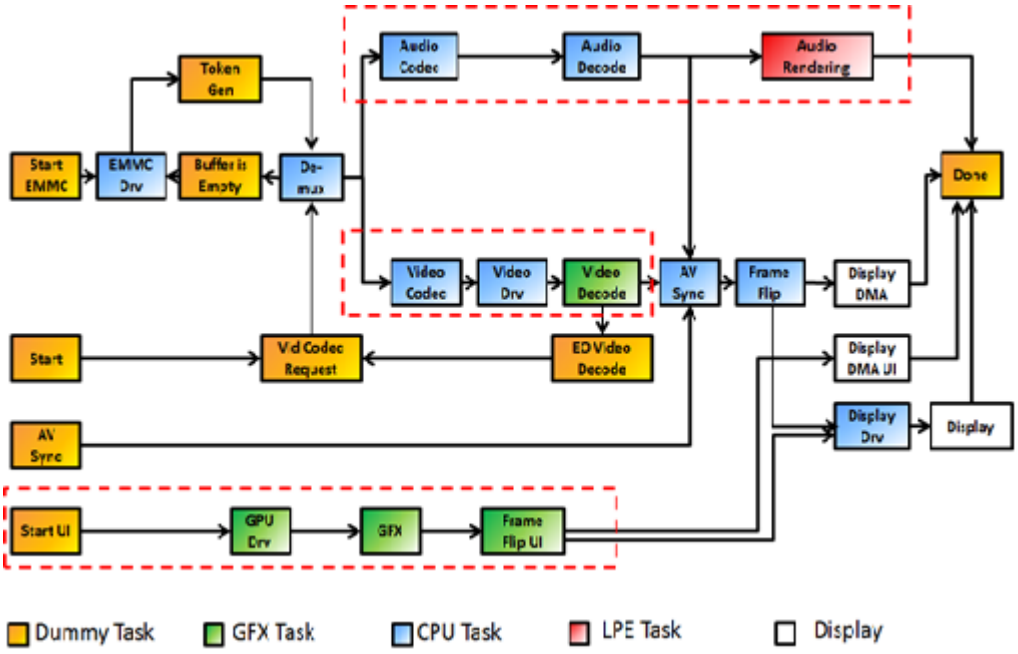

Figure 5.17: Generic Task Flow for Video Playback [16]

video decode. Thus task graph varies from platform to platform but the idea remains same and can easily be adopted for future platforms. Similarly specification for each task is platform dependent. IPC (Instruction per cycle) is different form platform to platform and hence the processing time. Also, HW gets optimized with the next generation platform and so specification changes. But modelling with different specification and dependencies to the past specification can be easily done with this task flow approach.

This approach also helps to simulate for different clip parameter like clip resolution, FPS, bit rate and different system configuration like display resolution, memory channels, MTs, CPU frequency, GPU frequency, etc. This is very useful for studying the effect of each parameter on the platform performance and power. This chapter presents for Video Playback use case but can be extended to video capture, HDMI, Wi-Di display, etc. Thus simulation of such real world use case helps to analyse the power and performance of a platform and also helps for scalability studies. It is possible to simulate with any workload parameter and system configuration and

analysis the behaviour of platform. The simulation approach saves time, cost and helps in optimization. It also helps for predicting power and performance values and in optimization for future platform where silicon is not available for measurement.

5.5 Simulation and Results

Video Playback is modelled using the task flow approach as discussed in section 5.1 and 5.4. It is however, measured on HW also for comparison. Correlation for throughput (BW in MB/s) of simulation with actual measurement is presented in Table III.
Error between measured and simulated is calculated as

$$Error\% = (|Measured - Simulated|)/Measured * 100\% \tag{5.1}$$

The correlation is done for 8 different clips and Table III shows that the BW correlation verified for 8 different clips and error for CPU BW is within 6%, GFX is within 8%, Display is within 2% and DDR in within 7%. Considering details of Read and Write BW, error is within 14%.These errors are within the acceptable range as measurement results vary within 10% range. Similarly even the residencies of the IPs are also measured and correlation with simulation result is as shown in Table IV.

The residency correlation in Table IV shows an error of less than 5% for CPU, less than 2% for GFX and less than 10% for media except for one clip that is due to measurement variation.
Correlation results for different CPU frequency are also presented. CPU frequency, do not affect the throughput but affect the CPU residency. Table V shows the correlation for 3 frequency points for 8 different clips. For all the combination, error is within 10%.

Clip Parameters	**1080p, 10Mbps, 30 FPS**	**1080p, 20Mbps, 30 FPS**	**1080p, 40Mbps, 30 FPS**	**1080p, 10Mbps, 60 FPS**	**1080p, 20Mbps, 60 FPS**	**1080p, 40Mbps, 60 FPS**	**720p, 10Mbps, 30 FPS**	**720p, 10Mbps, 60 FPS**
	%Error	**%Error**	**%Error**	**%Error**	**%Error**	**%Error**	**%Error**	**%Error**
DDR BW	4.2%	4.4%	5.2%	6.8%	4.9%	6.0%	3.8%	6.3%
DDR BW - Read	5.3%	5.6%	6.8%	8.6%	8.8%	8.4%	2.0%	3.6%
DDR BW - Write	1.8%	1.8%	2.0%	3.6%	2.0%	1.7%	8.3%	11.5%
Disp BW	1.9%	1.9%	1.9%	1.9%	1.9%	1.9%	1.9%	1.9%
GFX BW	4.9%	4.9%	4.9%	7.6%	7.9%	7.6%	0.9%	2.9%
GFX BW - Read	7.8%	7.9%	8.4%	12.0%	12.7%	12.6%	6.6%	6.6%
GFX BW - Write	1.1%	1.0%	0.3%	1.9%	1.8%	1.2%	8.8%	13.4%
CPU BW	4.7%	3.3%	5.7%	4.1%	2.9%	1.3%	5.7%	2.4%
CPU BW - Read	5.6%	3.7%	5.1%	6.2%	4.6%	1.9%	7.0%	5.2%
CPU BW - Write	1.4%	2.2%	7.6%	3.0%	2.3%	0.5%	1.8%	6.9%

Table III: Simulation v/s Measurement Correlation of BW

Clip Parameters	% Error	% Error	% Error
	CPU Residency	Media Residency	GFX Residency
1080p, 10Mbps,30 FPS	4.3%	0.0%	1.0%
1080p, 20Mbps, 30 FPS	1.5%	1.3%	1.0%
1080p, 40Mbps, 30 FPS	1.9%	5.7%	1.0%
1080p, 10Mbps, 60 FPS	1.3%	6.6%	0.0%
1080p, 20Mbps, 60 FPS	0.5%	7.8%	0.0%
1080p, 40Mbps, 60 FPS	0.8%	5.1%	0.0%
720p, 10Mbps, 30 FPS	9.1%	1.5%	1.0%
720p, 10Mbps, 60 FPS	0.7%	13.1%	2.0%

Table IV: Simulation v/s Measurement Correlation of Residency

Clip Parameters	% Error	% Error	% Error
	CPU freq - 532 MHz	CPU freq - 655 MHz	CPU freq - 931 MHz
1080p, 10Mbps,30 FPS	8.2%	6.0%	8.4%
1080p, 20Mbps, 30 FPS	1.3%	1.3%	2.6%
1080p, 40Mbps, 30 FPS	9.4%	11.9%	9.6%
1080p, 10Mbps, 60 FPS	4.1%	4.0%	4.7%
1080p, 20Mbps, 60 FPS	0.3%	0.0%	1.0%
1080p, 40Mbps, 60 FPS	4.4%	6.8%	4.6%
720p, 10Mbps, 30 FPS	8.5%	6.7%	7.5%
720p, 10Mbps, 60 FPS	3.9%	0.3%	9.3%

Table V: Simulation v/s Measurement Correlation of CPU Residency at Different CPU Frequency

5.6 Summary and Conclusion

PA Tool enables modelling a system level use-case using task graph approach. Such a system level task flow approach and tool would be very useful to assess system scalability of both HW and SW, as large number of derivative SOCs and Platforms are coming up to meet specific Customer needs. Task Flow models helps to establish common language between SOC, Platform Architects, SW developers and Validation Engineers and enable significant improvement in platform definition and HW/SW development. This approach is a significant step in studying Use Cases at a much higher abstraction than cycle accurate models and thus enabling to take Systems approach for Power/Performance efficiency - an absolute need of the hour to win Phone/Tablet segments. Also, as there are no limitations on simulation for memory, display resolution or clip FPS, it allows scalability studies across different platforms configuration and simulate cases that may not be possible on current system.

Chapter 6

Conclusion and Future Scope

6.1 Conclusion

This report is about Android workload characterization in silicon and simulation. First part of report deals with workload characterization on silicon and suggests methodologies for optimization like about choosing proper LFM and display resolution for next generation platform. Second frequency point is more optimum as compare to the lowest frequency at which today all CPU cores are working. Also, to be competitive it is better to use optimum display resolution. 1920x1080 pixel is the best resolution to choose which gives better performance. Second part of the report is about modelling and simulation of video playback and correlation with silicon measurement. This project explores that Task Flow approach and PA Tool helps to model a complex environment like smart phone and simulate a system level use-case. It present a video playback use-case and shows that in IPs, BW and residency correlate the measurement results with an error under accepatable limit. The error is mainly due to variation in measurent numbers. This also shows that various features can be added to the model like CPU frequncy effect on CPU residency, effect of GFX frequency, etc. This project shows a scope of platform and use-case modelling at much higher abstraction layer.

6.2 Future Scope

Exploring different methods to optimize power, performance and expanding features, uses of PA Tool :

- Optimization of platform/ suggestion for future platform in terms of better performance and low power consumption. One such experiment is to check power/performance improvement by limiting the P-State.
- Study scalability for different configurations for memory and display resolution and correlate with silicon
- Add features like GFX frequency effect, DDR Self Refresh, CPU Core and Package C_0 Residency and correlate with the silicon
- Add new use-cases like video playback with external HDMI connected and Video Record
- Extend model to core platforms and to future platforms

References

[1] CPU Governor overview, *https://www.kernel.org/doc/Documentation/cpu-freq/governors.txt*

[2] Sheran A. Gunasekera. "Android Apps Security, ISBN 978-1-4302-4062-4, Apress Publication. pp 1- 7.

[3] R. M. Ramanathan, Mary Doylt, "Intel Platforms Innovation Beyond Processors, Intel Software Library, March,2012.

[4] Clover Trail Plus SOC, *http://arstechnica.com/gadgets/2013/02/intel-gets-aggressive-with-new-smartphone-and-tablet-chips*

[5] Mohamed Shalam and Dina El-Sissy, "Online Power Management using DVFS for RTOS, Design and Test Workshop (IDT), IEEE Conference, 2009.

[6] Intel Corporation, "Intel Atom Processor Z5xx Series Datasheet, June 2010.

[7] Intel Corporation, "Power Use Case Analysis Document (UCAD), Revision 0.8, June 19th,2013.

[8] CPU C States description, *http://www.hardwaresecrets.com/printpage/Everything-You-Need-to-Know-About-the-CPU-C-States-Power-Saving-Modes/611*

[9] CPU P States description, *https://software.intel.com/en-us/blogs/2008/05/29/what-exactly-is-a-p-state-pt-1*

[10] Yan Gu, Samartjit Chakraborty and Wei Tsang Ooi, "Games are up for DVFS Design Automation Conference, IEEE, July 24-28, 2006, San Fransisco, California, USA.

[11] James Lapalme, Bart theelen, Nikolay Stoimenov, Jeroen Voeten, Lathar Thiele, El Mostapha Aboulhamid, "Y-Chart Based System Design: A Discussion on Approaches, ACM Journal, Vol 5, March 2009.

[12] Jari Kreku, Jani Penttila, Janne Kangas, Juha-Pekka Soininen. "Workload Simulation Methods for Evalution of Application Feasibility in a Mobile Multiprocessor Platform, IEEE Comuputer Society, Proceedings of the EUROMICRO Systems on Digital System Design, 2004.

[13] Subayal Khan, Eila Ovaska, Kari Tiensyrja, Jari Nurmi, "From Y-Chart to Seemless integration of Application Design and Perforamance Simulation, System on Chip, IEEE Conference, 2010.

[14] Intel Corporation,"ValleyView2 Primary Scalable Fabric (PSF), November 2012.

[15] Workload definition, *http://www.techopedia.com/definition/13544/workload*

[16] Intel Corporation, "Video Playback Task Flow Graph for BYT, October 2012.