UCI SAMSUNG PROJECT

Phase 3.2 Deliverable:
Final report on mobile GPU characterization and opportunities for power and performance improvement

Period of Activity: 09/15/2015 - 01/14/2016
Due Date: 01/14/2016

University of California, Irvine
Principal Investigator: Nikil Dutt, dutt@ics.uci.edu
Graduate Student Researcher: Jurngyu Park, jurngyup@ics.uci.edu
I Correlation between micro-benchmarks and Sample Games
I-A Definition of Workload ................................................. 3
I-B Low/High Thresholds for CPU/GPU/Memory Workloads ............... 4
I-C Classification of micro-benchmarks .................................... 4
I-D Correlation to Sample Mobile Games .................................. 5

II Estimation of Improvements/Outline of Opportunities ................. 5
II-A Priority 1: HLH and HHH .............................................. 6
II-B Priority 2: HLL and HHL .............................................. 7
II-C Priority 3: LLL, LLH, LHL, and LHH ................................. 7
II-D Outline of Opportunities and Heuristic min/max Frequency Selection .......... 8

III A Sample Policy Design .................................................. 8

IV Evaluation of micro-benchmarks and Sample Games ............... 9
IV-A Experimental Setup .................................................... 9
IV-B Results of micro-benchmarks .......................................... 10
IV-C Results of Sample Games ............................................. 11

V Portability Issue (Using the Proposed Methodology on a New Platform) 12

VI Conclusion .................................................................. 14

References ....................................................................... 14
Phase 3.2: Final report on mobile GPU characterization and opportunities for power and performance improvement

In the previous phase (Phase 3.1), we executed the micro-benchmarks on the target platform; studied the correlations between the workload variation and power/performance (FPS)/energy per frame (EpF) metrics based on the experimental results, in order to estimate improvements in power and performance efficiency for energy-efficient policy design.

And then, as shown in Figure 1, this phase (Phase 3.2) correlates the micro-benchmarks to sample graphics-intensive applications (e.g., mobile games) to estimate improvements in power and performance efficiency; outline opportunities for synergistic CPU-GPU memory governors optimizations for mobile DVFS design; propose a sample integrated CPU-GPU DVFS policy for quality-aware energy savings; and evaluate the sample policy using micro-benchmarks and sample games.

I. CORRELATION BETWEEN MICRO-BENCHMARKS AND SAMPLE GAMES

A. Definition of Workload

In order to correlate the workload of micro-benchmarks to sample mobile games, the definition of CPU/GPU/Memory workloads is needed. These workload variations are typically quantified using a cost metric that is a product of the utilization and frequency [1] [2]. Accordingly, we deploy normalized CPU- and GPU costs as shown in Equation(1), where the cost is defined as the product of the processor current average utilization and its current average frequency divided by the product of the maximum utilization and its maximum frequency.

\[
\text{Normalized Cost} = \frac{\text{Curr Util.} \times \text{Curr Freq.}}{\text{Max Util.} \times \text{Max Freq.}} 
\]

For CPU utilization, the highest CPU utilization among CPU cores is used according to the assumption that there is usually one graphics rendering thread mainly affecting the graphics performance for most mobile graphics applications, and the utilization of the thread is mostly highest among threads.

We also characterize the correlation between the normalized cost function (Cost) and power consumption of each component (CPU/GPU/memory). Figure 2 plots the average power of CPU/GPU/memory power consumption and average CPU/GPU/memory Cost for the 125 combinations of micro-benchmarks using five different workloads of mb-App, mb-VerM and mb-TexM (5x5x5=125). We observe near-linear increase in average power with the increasing Cost. Therefore, we assume that CPU/GPU/memory workloads can be defined by the normalized CPU/GPU/Memory cost function.
B. Low/High Thresholds for CPU/GPU/Memory Workloads

After the definition of workload, workload estimation of micro-benchmarks by using the CPU/GPU/memory Costs can be feasible. However, in order to differentiate (detect) low or high CPU/GPU/memory workloads, we should define the thresholds of utilization and frequency of each component.

For the threshold of utilization, we investigate scale-up thresholds in default governors (i.e., CPU interactive governor, GPU ARM’s Mali Midgard governor, and memory simpleexynos governor). We assume that these thresholds defined in the default governors are important indicators to scale up frequencies; the thresholds are considered as high utilization state at that moment. According to our investigation, the thresholds are like these: CPU scale-up threshold (the default target loads = 90), GPU scale-up threshold (max_threshold = 90 except 480Mhz and 543Mhz), and memory scale-up threshold (DFE_UPTHRESHOLD = 60). And then, we use the mean (average) of minimum and maximum frequency shown in Table I. Finally, we can get the Low/High detection thresholds for CPU/GPU/memory workloads using Equation (2): 72% for CPU, 60% for GPU, and 53 for Memory. (We summarize CPU/GPU/Memory Low/High workload detection thresholds shown in Table I).

\[
\text{Low/High Threshold} = \frac{(\text{Scale-up Threshold}) \times (\text{Mean of min and max Freq.})}{\text{Max_Util.} \times \text{Max_Freq.}}
\]  

(2)

TABLE I: Summary of CPU/GPU/Memory Low/High Thresholds related information

<table>
<thead>
<tr>
<th>component</th>
<th>scale-up th.</th>
<th>min Freq.</th>
<th>max Freq.</th>
<th>max Util.</th>
<th>Low/High th.</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>90</td>
<td>1.2 Ghz</td>
<td>2.0 Ghz</td>
<td>100%</td>
<td>72</td>
</tr>
<tr>
<td>GPU</td>
<td>90</td>
<td>177 Mhz</td>
<td>543 Mhz</td>
<td>100%</td>
<td>60</td>
</tr>
<tr>
<td>Memory</td>
<td>60</td>
<td>633 Mhz</td>
<td>825 Mhz</td>
<td>100%</td>
<td>53</td>
</tr>
</tbody>
</table>

C. Classification of micro-benchmarks

As shown in Figure 3, combinations of micro-benchmarks can be classified with 8 cases using the thresholds: LLL, LLH, LHL, LHH, HLL, HLH, HHL, and HHH. And, the representative micro-benchmark in each case and the corresponding CPU, GPU, and memory Costs are shown additionally.
D. Correlation to Sample Mobile Games

In addition, we correlate the classified micro-benchmarks to sample graphics-intensive applications. Among many different types of games, the representative sample games and the corresponding CPU/GPU/memory costs are shown in Figure 4. Unlike micro-benchmarks, we could not observe each corresponding game in each case. Therefore, some sample games are classified relatively (the red and orange cases).

![Fig. 4: Correlation to Sample Games.](image)

In this section, after the definition of CPU/GPU/memory workloads and the description of Low/High detection thresholds for the workloads, we classified the micro-benchmarks and correlate them to sample games. In the next section, using the eight cases of the representative micro-benchmarks, we will estimate improvements of each case; and outline opportunities of synergistic CPU-GPU-memory governors optimizations for mobile DVFS design.

II. ESTIMATION OF IMPROVEMENTS/OUTLINE OF OPPORTUNITIES

Through the estimation of improvements in each case, we outline the opportunities for CPU and GPU governors design. We describe the cases according to priority (high priority in high opportunities of energy saving with minimal FPS degradation): 1) priority 1 - HLH and HHH cases. 2) priority 2 - HLL and HHL cases. 3) priority 3 - LLL, LLH, LHL, and LHH cases.
A. Priority 1: HLH and HHH

Figure 5.(a) summarizes the opportunities of HLH and HHH cases in CPU and GPU governors. And, the representative micro-benchmarks for HLH and HHH cases (Figure 5.(b)) correspond to the Figure 14 (GPU governor perspective) and 15 (CPU governor perspective) of mb-App low-med + mb-VerM low-med + mb-TexM med-high and the Figure 12 (GPU governor perspective) and 13 (CPU governor perspective) of mb-App low-med + mb-VerM med + mb-TexM med respectively in the phase 3.1 report.

First of all, in the estimation of improvements from the perspective of CPU governor in the HLH case, FPS is almost similar but power consumption is proportional to CPU frequency change (high power in high CPU frequency). In particular, the consumed power of the default governors is almost similar to power consumption of the maximum CPU frequency. It clearly means that lower CPU frequency than the default CPU governor is better for energy saving without significant FPS degradation for the HLH case.

According to our speculation, the HLH is CPU dominant (high CPU cost and low GPU cost) and the default CPU governor does not consider high memory cost but scales frequency based on only CPU utilization. However, high memory cost may increase memory access latency time, which was also described in the paper [3]. If the memory access latency time is included in the CPU busy time, the CPU utilization will increase, which also results in CPU frequency scale-up and increase of power consumption without additional FPS improvement. Therefore, lower CPU frequency is an opportunity in the HLH case for energy saving without significant FPS degradation. Of course, how much lower CPU frequency is also important point, but in this project, this value will be selected heuristically/empirically with feasibility tests using many different combinations of micro-benchmarks.

Second, from the perspective of GPU governor in the HLH case, FPS is almost similar except the minimum GPU frequency (177Mhz); lower GPU frequency has lower power consumption; and power consumption of the default is almost similar to that of 350Mhz. (According to our speculation, in the CPU-dominant workload of the HLH case, GPU frequency change does not have significant effects on FPS). Therefore, lower GPU frequency (greater than 177 Mhz) has energy saving without significant FPS drop.

Third, from the perspective of CPU governor in the HHH case, FPS is almost same and power is proportional to CPU frequency (the default also has very low power and low CPU frequency). Therefore, lower CPU frequency is better for the HHH case if there is no significant FPS degradation.

Finally, from the perspective of GPU governor in the HHH case, FPS is almost similar if GPU frequency is greater than 266Mhz, which means that a proper GPU frequency (350Mhz in the example combination of the Figure 13 in the phase 3.1) can have energy saving without significant FPS drop.

Note that how much lower or proper frequencies are also important points, but in this project for a sample policy, this values will be selected heuristically/empirically with feasibility tests using many different combinations of micro-benchmarks.
B. Priority 2: HLL and HHL

Figure 6.(a) summarizes the opportunities of HLL and HHL cases in CPU and GPU governors. And, the representative micro-benchmarks for HLL and HHL cases (Figure 6.(b)) correspond to \textit{mb-App med} and to \textit{mb-App med + mb-VerM med-high} respectively (shown in page 5 and 9 of the phase 3.1 report).

First, from the perspective of CPU governor in the HLL and HHL cases, power consumption of 2.0Ghz (the maximum) is extraordinarily higher compared to 1.8Ghz. (FPS is 11% lower but power consumption is 34% lower in the representative micro-benchmark than 1.8Ghz). (According to our speculation, the default CPU governor is cluster-based INTERACTIVE governor, but one main rendering process of graphics applications is mainly used on a single core. In addition, static configurations between CPU frequency/voltage and memory frequency/voltage also may result in high power consumption; high CPU frequency is fixed to high CPU voltage, high memory frequency, and high memory voltage by static configuration). Therefore, after the feasibility tests, we will adjust the maximum CPU frequency.

---

<table>
<thead>
<tr>
<th>Prio.</th>
<th>Case</th>
<th>Governor</th>
<th>Opportunities</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>HLL</td>
<td>CPU</td>
<td>Power of 2000Mhz is extraordinary higher</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GPU</td>
<td>480Mhz &amp; 543Mhz are better</td>
</tr>
<tr>
<td></td>
<td>HHL</td>
<td>CPU</td>
<td>Power of 2000Mhz is extraordinary higher</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GPU</td>
<td>The default is better</td>
</tr>
</tbody>
</table>

---

**Cost Level** | **Representative Micro-benchmark** | **Cost in Phase 3.1**
--- | --- | ---
HLL | App 2-4 | 96 17 18 | Page 5
HHL | App 2 + Ver 3-4 | 77 86 20 | Page 9

(a) Opportunities in Governors (b) Representative Micro-benchmark

Fig. 6: Estimation of Improvements: Priority 2 - HLL and HHL Cases.

Second, from the perspective of GPU governor in the HLL case, the two highest GPU frequencies (480Mhz and 543Mhz) are better for energy saving (7% FPS drop but 38% energy saving in the representative micro-benchmark) in the HLL case (CPU-dominant workload). According to our investigation, when 480Mhz and 543Mhz are selected, CPU frequency is capped to CPU 1.6Ghz in the default GPU governor. (Alternatively, if there is significant FPS drop in the feasibility tests, we could adopt one of lower GPU frequencies than 480Mhz, which is not capped to 1.6Ghz).

Finally, from the perspective of GPU governor in the HHL case, the default GPU governor has the best results in terms of FPS and power consumption. Therefore, we apply the scheme of the default GPU governor as a GPU governor policy for the HLL case.

C. Priority 3: LLL, LLH, LHL, and LHH

---

<table>
<thead>
<tr>
<th>Prio.</th>
<th>Case</th>
<th>Governor</th>
<th>Opportunities</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>LLL, LHH</td>
<td>CPU</td>
<td>Lower CPU frequency is better</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GPU</td>
<td>Lower GPU frequency is better</td>
</tr>
<tr>
<td></td>
<td>LHL, LHH</td>
<td>CPU</td>
<td>Lower CPU frequency is better</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GPU</td>
<td>The default is better</td>
</tr>
</tbody>
</table>

---

**Cost Level** | **Representative Micro-benchmark** | **Cost in Phase 3.1**
--- | --- | ---
LLL | App0, VerM0, TexM0 | - - - | Page 4
LLH | TexM2~4 | - - - | Page 6
LHL | VerM 2~4 | - - - | Page 4
LHH | VerM2 + TexM2 + App0 | - - - | Page 11

(a) Opportunities in Governors (b) Representative Micro-benchmark

Fig. 7: Estimation of Improvements: Priority 3 - LLL, LLH, LHL, and LHH Cases.

Figure 7.(a) summarizes the opportunities of LLL, LLH, LHL, and LHH cases in CPU and GPU governors, and Figure 7.(b) shows the corresponding representative micro-benchmarks.
From the perspective of CPU and GPU governors in the LLL and LLH cases, lower CPU and GPU frequencies are better for energy saving without significant FPS degradation. (For some combinations of micro-benchmarks, the minimum CPU and GPU frequencies are still high enough for the LLL and LLH cases).

Moreover, from the perspective of CPU and GPU governors in the LHL and LHH cases, because these cases are GPU-dominant workloads, FPS and power consumption are proportional to GPU frequency, but CPU frequency change does not have significant effects on FPS and power consumption. Therefore, for energy saving without significant FPS drop, lower CPU frequency and the default GPU governor scheme are better for these cases.

D. Outline of Opportunities and Heuristic min/max Frequency Selection

For heuristic selection of frequency ranges for a sample policy design in addition to outlining the opportunities, 60 combinations of micro-benchmarks for 8 cases are used. (the detailed combinations of micro-benchmarks are shown in Figure 10).

And, Figure 8 summarizes CPU/GPU minimum/maximum frequency ranges for energy saving with minimal FPS degradation, which were chosen heuristically/empirically from the opportunities and comprehensive feasibility tests. (The reason we do not consider memory capping in this project is that memory DVFS (or capping) is not effective in terms of power saving without FPS degradation. [2])

<table>
<thead>
<tr>
<th>Prio</th>
<th>Case</th>
<th>Governor</th>
<th>Opportunities</th>
<th>min/max Frequency Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>HLH</td>
<td>CPU</td>
<td>Lower CPU frequency is better.</td>
<td>1.2 ~ 1.5 Ghz</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GPU</td>
<td>Lower GPU frequencies (&gt;177Mhz) are better.</td>
<td>266 ~ 350 Mhz</td>
</tr>
<tr>
<td></td>
<td>HHH</td>
<td>CPU</td>
<td>Lower CPU frequency is better.</td>
<td>1.2 ~ 1.5 Ghz</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GPU</td>
<td>350Mhz GPU frequency is better in terms of Power and EpF.</td>
<td>350 ~ 480 Mhz</td>
</tr>
<tr>
<td>2</td>
<td>HLL</td>
<td>CPU</td>
<td>Power of 2000Mhz is extraordinary higher</td>
<td>1.2 ~ 1.9 Ghz</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GPU</td>
<td>480Mhz &amp; 543Mhz are better</td>
<td>420 ~ 543 Mhz</td>
</tr>
<tr>
<td></td>
<td>HHL</td>
<td>CPU</td>
<td>Power of 2000Mhz is extraordinary higher</td>
<td>1.2 ~ 1.9 Ghz</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GPU</td>
<td>The default is better</td>
<td>177 ~ 543 Mhz</td>
</tr>
<tr>
<td>3</td>
<td>LLL</td>
<td>CPU</td>
<td>Lower CPU frequency is better</td>
<td>1.2 ~ 1.5 Ghz</td>
</tr>
<tr>
<td></td>
<td>LLH</td>
<td>GPU</td>
<td>Lower GPU frequency is better</td>
<td>177 ~ 350 Mhz</td>
</tr>
<tr>
<td></td>
<td>LHL</td>
<td>CPU</td>
<td>Lower CPU frequency is better</td>
<td>1.2 ~ 1.5 Ghz</td>
</tr>
<tr>
<td></td>
<td>LHH</td>
<td>GPU</td>
<td>The default is better</td>
<td>177 ~ 543 Mhz</td>
</tr>
</tbody>
</table>

Fig. 8: Opportunities and Heuristic Frequency Selection.

III. A Sample Policy Design

We propose a sample policy for CPU-GPU-memory Cost-aware synergistic DVFS, where CPU and GPU minimum/maximum frequency capping technique is applied selectively. We base our heuristic on the observation that characterization (Figure 8) of CPU-GPU-memory cost correlation is a good indicator of graphics application workloads. Figure 9 has the diagram of the finite state machine (fsm) representing our model, with switching conditions as annotations on the edges.

Our proposed heuristic consists of 3 states named CPU-GPU-Memory Cost Calculation (CGMCC), Synergistic Optimization (SO), and Temp Turbo Mode (TTM), where:
• CPU-GPU-Memory Cost Calculation (CGMCC): data capturing and cost calculation at the start of a epoch window. No min/max frequency setting (same with the previous setting).

• Synergistic Optimization (SO): appropriate min/max frequency range is selected. CPU frequency ranges from 1.2Ghz to 1.9Ghz, and GPU frequency ranges from 177Mhz to 543Mhz.

• Temp Turbo Mode (TTM): one level higher CPU (two level higher GPU) maximum frequency than the SO state.

Fig. 9: A Sample Policy: Finite State Machine (FSM) implemented by our policy.

We partition the execution time of applications into 100ms epoch windows. At the start of each epoch window, our finite state machine calculate CPU, GPU, and memory Costs by capturing the average frequency and the highest average utilization. We use the CPU Cost of 72%, the GPU Cost of 60, and the memory Cost of 53 as the thresholds for determining each case (8 cases from LLL to HHH), as described in Section I-B. Detailed description of our heuristic follows.

The FSM is initialized to the CGMCC state, at the start of the epoch window (100ms), CPU, GPU, and memory costs are calculated after capturing the frequency and the utilization from each governor, then the FSM transitions to the SO state or the TTM state. If the CPU or GPU maximum utilization counter ($CPU_{max\ util\ cnt}$ or $GPU_{max\ util\ cnt}$) is greater than 2, then the FSM transitions to the TTM state. Otherwise, our heuristic performs the synergistic optimization using DVFS (i.e., using the min/max frequency capping technique).

During the SO state the heuristic selects the new min/max frequency setting using Figure 8, which was characterized from the opportunities and feasibility tests in the previous section. Otherwise, during the TTM state one level higher CPU maximum frequency and two level higher GPU maximum frequency (under the upper-bound CPU (1.9Ghz) and GPU (543Mhz)) compared to the SO state, are selected temporarily to prevent unpredicted FPS drop under continuous maximum utilization. The SO (or TTM) after the CGMCC is performed once per epoch window, at the start of the epoch.

IV. EVALUATION OF MICR0-BENCHMARKS AND SAMPLE GAMES

A. Experimental Setup

We evaluate the sample policy using min/max capping technique on the ODROID-XU3 development board installed with Android 4.4.2 and Linux 3.10.9; Table II summarizes our platform configurations.
TABLE II: Platform Configuration

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Device</td>
<td>ODROID-XU3</td>
</tr>
<tr>
<td>SoC</td>
<td>Samsung Exynos5422</td>
</tr>
<tr>
<td>CPU</td>
<td>Cortex-A15 2.0Ghz and Cortex-A7 Octa-core CPUs</td>
</tr>
<tr>
<td>GPU</td>
<td>Mali-T628 MP6, 543Mhz</td>
</tr>
<tr>
<td>System RAM</td>
<td>2Gbyte LPDDR3 RAM at 933MHz</td>
</tr>
<tr>
<td>Mem. Bandwidth</td>
<td>up to 14.9GB/s</td>
</tr>
<tr>
<td>OS(Platform)</td>
<td>Android 4.4.2</td>
</tr>
<tr>
<td>Linux Kernel</td>
<td>3.10.9</td>
</tr>
</tbody>
</table>

The ODROID platform is equipped with four TI INA231 power sensors measuring the power consumption of big CPU cluster (CPU-bc), little CPU cluster (CPU-lc), GPU and memory respectively. The CPU supports cluster-based DVFS at nine frequency levels (from 1.2Ghz to 2.0Ghz) in CPU-bc and at seven frequency levels (from 1.0Ghz to 1.6Ghz) in CPU-lc, and GPU supports six frequency levels (from 177Mhz to 543Mhz).

**Benchmark Sets:** we use 60 combinations of micro-benchmarks (Figure 10.(a)) and 14 sample games (Figure 10.(b)) to evaluate the sample policy. (The numbers of micro-benchmarks were heuristically/empirically chosen and some sample game cases such as LHL, LLH, HHL and HLH are relatively classified.)

![Fig. 10: Benchmark Sets.](a) Combinations of micro-benchmarks  
(b) Sample Games

**B. Results of micro-benchmarks**

The sample policy achieves EpF improvement of 7.7% and negligible FPS decline (1.7%) on average for 60 micro-benchmarks. Figure 11.(a) shows the results on FPS, power consumption, and EpF of low CPU cost micro-benchmarks. The results are almost similar to the default (0.4% FPS drop and 1.5% energy saving). However, high CPU cost micro-benchmarks (Figure 11.(b)) achieve significant EpF improvement of 13.9% with little FPS degradation of 3.1% on average.

According to our speculation, GPU cores are dedicated only for graphics rendering tasks; the default GPU governor scales up the frequency step by step (conservatively). However, CPU-cores execute many different types of tasks in addition to graphics rendering tasks; the cluster-based (not core-based) default CPU INTERACTIVE governor scales up aggressively for general performance improvements.

For additional comparison, Figure 12.(a) shows the results of low GPU cost micro-benchmarks, which achieve EpF improvement of 9.9% with little FPS degradation of 2.1% on average. And, high GPU cost micro-benchmarks (Figure 11.(b)) achieve EpF improvement of 5.4% with little FPS degradation of 1.4% on average. (No significant difference, compared to Low CPU vs. High CPU Cost).
C. Results of Sample Games

Figure 13 shows the results of all sample games for 8 cases and the sample policy achieves EpF improvement of 11% and negligible FPS decline (1.8%) on average, (7.7% EpF improvement and 1.7% FPS drop for all 60 micro-benchmarks).
Figure 14.(a) shows the results on FPS, power consumption, and EpF of low CPU cost sample games. Energy saving is a little bit better than the default (almost same FPS and 3.7% energy saving). However, high CPU cost sample games (Figure 14.(b)) achieve significant EpF improvement of 18.2% with little FPS degradation of 3.7% on average.

![Fig. 14: Results of Sample Games (Low CPU vs. High CPU).](image)

For additional comparison, Figure 15.(a) shows the results of low GPU cost sample games, which achieve EpF improvement of 12.2% with little FPS degradation of 2.6% on average. And, high GPU cost sample games (Figure 11.(b)) achieve EpF improvement of 9.8% with little FPS degradation of 1.0% on average. (No significant difference, compared to Low CPU vs. High CPU Cost).

![Fig. 15: Results of Sample Games (Low GPU vs. High GPU).](image)

In summary, from the similar patterns of the results of micro-benchmarks and real sample games, the results clearly show that the sample policy based on the opportunities is able to achieve significant improvement in EpF with little FPS decline for micro-benchmarks and the correlated sample games on average.

V. PORTABILITY ISSUE (USING THE PROPOSED METHODOLOGY ON A NEW PLATFORM)

In this section, we deal with the portability issue. In other words, is it still applicable to use the proposed methodology on a new platform? To this end, we adopt a scenario study: platform change from Exynos 5422 (adopted in Galaxy S5) to Exynos 7420 (adopted in Galaxy S6) based platform. We assume that a platform is changed from the ODROID-XU3 board to the MV7420 board; the comparison of platform specifications is like Figure 16.
In order to apply our proposed methodology on a new platform, we should check both the differences of hardware and software on the new platform (MV7420) and availability of our abstract and logical pipeline models (abstract mobile CPU-GPU graphics pipeline model and OpenGL ES based logical graphics rendering pipeline model), which were described in the phase 1.

First of all, from the perspective of hardware, fabrication technology (28nm to 14 nm), number of cores, and CPU-GPU-memory minimum/maximum frequencies are mainly changed on a new platform. For these kinds of changes, our abstract mobile CPU-GPU graphics pipeline model is still applicable because the model focuses on Vertex fetching, Vertex shader programming, Texture fetching, and Fragment shader programming, which are independent on the changes (fabrication technology and number of cores). Additionally, for CPU-GPU-memory minimum/maximum frequency changes, it can be resolved by new measurements on a new platform.

Moreover, from the perspective of software, Android version or OpenGL ES libraries could be changed. For the changes, our OpenGL ES based logical graphics rendering pipeline model is still applicable because fundamental Client-Server based SW frameworks are still same in spite of Android version change and OpenGL ES 3.x is fully backwards compatible with OpenGL ES 2.0, in spite of the newer OpenGL ES version change. (In software changes, it would be mainly features/functions changes with backward compatibility).

<table>
<thead>
<tr>
<th>Feature</th>
<th>Current platform</th>
<th>A new platform</th>
</tr>
</thead>
<tbody>
<tr>
<td>Released</td>
<td>Q2 2014</td>
<td>Q2 2015</td>
</tr>
<tr>
<td>SoC (Fab)</td>
<td>Exynos 5422 (28 nm)</td>
<td>Exynos 7420 (14 nm)</td>
</tr>
<tr>
<td>Device</td>
<td>ODROID XU3</td>
<td>MV7420</td>
</tr>
<tr>
<td>CPU</td>
<td>Octa (Cortex-A14 2.0Ghz Quad + Cortex A7 1.6Ghz Quad)</td>
<td>Octa (Cortex-A57 2.1Ghz Quad + Cortex A53 1.5Ghz Quad)</td>
</tr>
<tr>
<td>GPU</td>
<td>Mali-T628 MP6, 543Mhz</td>
<td>Mali-T760 MP8, 700 Mhz</td>
</tr>
<tr>
<td>System RAM</td>
<td>2Gbyte LPDDR3 at 825Mhz</td>
<td>3Gbyte LPDDR4 at 1600 Mhz</td>
</tr>
<tr>
<td>Mem. Bandwidth</td>
<td>Up to 14.9 GB/s</td>
<td>Up to 25.6 GB/s</td>
</tr>
<tr>
<td>Android</td>
<td>Android 4.4.2</td>
<td>Android 5.1 (Lollipop)</td>
</tr>
<tr>
<td>Linux Kernel</td>
<td>3.10.9</td>
<td>3.10x</td>
</tr>
</tbody>
</table>

Figure 16: A Scenario Study: Platform Change [4].

Figure 17 illustrates the overall road-map of our proposed methodology (from phase 1 to phase 3.2). If a few components could be considered (i.e., evaluated, modified, or measured on a new platform), the proposed methodology is still applicable for new platforms. With the assumption that abstract mobile CPU-GPU graphics pipeline model and OpenGL ES based logical graphics rendering pipeline model are still applicable, the red-outlined components should be considered on a new platform.

First, evaluation of micro-benchmarks on a new platform should be considered. In order to evaluate different graphics workloads (e.g., various ranges of 20 FPS - 60 FPS) on a new platform, stresses of each
stage (e.g., number of vertices) could/might be changed. Second, according to the minimum/maximum
CPU-GPU-memory frequency changes, measurements on a new platform should be done. Finally, heuristic
optimization in policy design also should be done with the feasibility tests using combinations of micro-
benchmarks.

VI. CONCLUSION

In this phase (phase 3.2), we correlate the micro-benchmarks to real sample games to estimate im-
provements in power and performance efficiency; outline opportunities for synergistic CPU-GPU memory
governors optimizations for mobile DVFS design; and propose a sample integrated CPU-GPU DVFS policy
for quality-aware energy savings.

We then evaluate the efficacy of the sample policy using the combinations of micro-benchmarks and the
sample games. Our experimental results on the ODROID platform show that the sample policy improves
energy per frame by 7.7% and 11.0% on average and achieves little FPS decline by 1.7% and 1.8% on
average for the 60 micro-benchmarks and the 14 sample games respectively, compared to the default CPU-
and GPU governors, with negligible overhead in execution time and power consumption. Finally, for the
portability of the proposed methodology on a new platform, we introduce the study based on the scenario
of the platform change.

REFERENCES