UCI SAMSUNG PROJECT

Phase 3.1 Deliverable:
Report on Experimental analyses of micro-benchmark execution

Period of Activity: 06/15/2015 - 09/15/2015
Due Date: 09/30/2015

University of California, Irvine
Principal Investigator: Nikil Dutt, dutt@ics.uci.edu
Graduate Student Researcher: Jurngyu Park, jurngyup@ics.uci.edu
I Results of each micro-benchmark 3
I-A Each micro-benchmark ........................................... 3
I-A1 mb-VerM ......................................................... 3
I-A2 mb-App ......................................................... 4
I-A3 mb-TexM ......................................................... 5
I-A4 mb-VerSh and mb-FragSh ........................................ 6

II Results of combinations of the micro-benchmarks 8
II-A Combinations of two micro-benchmarks .................. 8
II-A1 mb-App med + mb-VerM change .......................... 8
II-A2 mb-App med + mb-TexM change .......................... 9
II-A3 mb-App high + mb-VerM change and mb-App high + mb-TexM change 10
II-B Combinations of three micro-benchmarks ................ 11
II-B1 mb-VerM med + mb-TexM med + mb-App change ........ 11
II-B2 mb-VerM low-med + mb-TexM med-high + mb-App .......... 12

III Summary and Next Phase 13
III-A Summary of each micro-benchmark ....................... 13
III-B Summary of two micro-benchmarks .......................... 14
III-C Summary of three micro-benchmarks ....................... 14
III-D For the Next Phase .............................................. 14

IV APPENDICES 15
IV-A APPENDIX A ....................................................... 15
IV-B APPENDIX B ....................................................... 17
Phase 3.1: Report on Experimental analyses of micro-benchmark execution

In this phase, we execute the micro-benchmarks on the target mobile platform, ODROID-XU3, and study the correlations between the platform hardware characteristics (stresses of each stage) and power / performance (FPS) / energy per frame (EpF) metrics based on the experimental results in order to observe and estimate improvements in power and performance efficiency; and design energy-efficient policy.

In the execution of micro-benchmarks, in order to analyze systematically the effects of each micro-benchmark we designed measurements with two steps: 1) execution of each micro-benchmark and 2) execution of combinations of micro-benchmarks.

I. RESULTS OF EACH MICRO-BENCHMARK

TABLE I: Micro-benchmarks and their Pipelined Workloads

<table>
<thead>
<tr>
<th>MBs</th>
<th>CPU</th>
<th>Vertex Fetch</th>
<th>Vertex Shaders</th>
<th>Texture Fetch</th>
<th>Fragment Shaders</th>
</tr>
</thead>
<tbody>
<tr>
<td>mb-VerM</td>
<td>Only GL API</td>
<td>Num. of Ver.</td>
<td>Minimized</td>
<td>Minimized</td>
<td>Minimized</td>
</tr>
<tr>
<td>mb-TexM</td>
<td>Only GL API</td>
<td>Minimized</td>
<td>Minimized</td>
<td>Texture Img. Size</td>
<td>Minimized</td>
</tr>
<tr>
<td>mb-VerSh</td>
<td>Only GL API</td>
<td>Minimized</td>
<td>Vertex Sh. Prog.</td>
<td>None</td>
<td>Minimized</td>
</tr>
<tr>
<td>mb-FragSh</td>
<td>Only GL API</td>
<td>Minimized</td>
<td>Minimized</td>
<td>None</td>
<td>Frag. Sh. Prog.</td>
</tr>
<tr>
<td>mb-App</td>
<td>GL API + Loop</td>
<td>Minimized</td>
<td>Minimized</td>
<td>None</td>
<td>Minimized</td>
</tr>
</tbody>
</table>

A. Each micro-benchmark

In the execution of each micro-benchmark, we measured FPS, power, and energy per frame (EpF) at different frequencies of GPU, big CPU cluster (CPU-bc), and little CPU cluster (CPU-lc). However, the effects of CPU-lc frequencies on FPS, power, and EpF were little (within 5%) compared to the default governors. Therefore, we focus on the effects of GPU and CPU-bc frequencies based on the assumption that impacts of CPU-lc could be negligible compared to those of GPU and CPU-bc for graphics applications on ODROID-XU3.

1) mb-VerM: As shown in Table I, the number of vertices was changed for the mb-VerM and we minimized stresses of other stages. The x-axis is the change of the number of vertices (e.g., 64 means 24 x 64 vertices), and the y-axis is FPS, Power (mW), and energy per frame (EpF) respectively as shown in Figure 1.
**a) GPU frequency effects**: As shown in Figure 1, FPS is inversely proportional to the increased number of vertices and proportional to GPU frequency change above 256 number of vertices (because of the maximum FPS of 60 by VSYNC enabling). Power is proportional to the increased number of vertices and proportional to GPU frequency change except 266Mhz. And, EpF is proportional to the increased number of vertices. And EpF is proportional to GPU frequency change below 128 (lower GPU frequency is better for EpF) and almost similar above 256 except 266Mhz and 128Mhz.

**Specific Observations**: As shown in Figure 1(b) and (c), Power and EpF of 266Mhz are extraordinarily higher than other frequencies for overall workloads.

![Fig. 1: Results of mb-VerM at different GPU frequencies](image)

**b) CPU-bc frequency effects**: As shown in Figure 2, FPS is inversely proportional to the increased number of vertices, but is not changed by CPU frequency change (because of very low CPU workload). Power is proportional to the increased number of vertices and proportional to CPU frequency change. And, EpF is proportional to the increased number of vertices and proportional to CPU frequency change.

**Specific Observations**: As shown in Figure 2(b), power consumption of 2000Mhz is extraordinary higher than other frequencies. And in Figure 2(b) and (c), Power and EpF of default CPU governor are always similar with those of 1200Mhz, because this test case has low CPU workload. Therefore, additionally Section II-A shows the results of medium CPU workload and high CPU workload with combined micro-benchmarks.

![Fig. 2: Results of mb-VerM at different CPU frequencies](image)

2) **mb-App**: As shown in Table I, CPU loop iterations (i.e., a configurable code stub that has only CPU workload like 'for' loop execution) were changed for mb-App and we minimized stresses of other stages with fixed GPU workloads. The x-axis is the change of the CPU workload level, and the y-axis is FPS, Power, and EpF respectively as shown in Figure 3.
a) **GPU frequency effects**: As shown in Figure 3, FPS is inversely proportional to the increased CPU workload above workload 2, but there is no significant FPS change by GPU frequency change. Power is proportional to the increased CPU workload only below workload 2, and almost similar above workload 2. And, EpF is proportional to the increased CPU workload.

**Specific Observations**: As shown in Figure 3(b) and (c), Power and EpF of high GPU frequency (480Mhz and 543Mhz) are lower than them of other frequencies above workload 2 (CPU dominant workloads) with slightly lower FPS. According to our observation (or speculation), higher GPU frequency and lower CPU frequency for above workload 2 compared to the default CPU- and GPU frequency could be beneficial for Power and EpF.

![Fig. 3: Results of mb-App at different GPU frequencies](image)

b) **CPU-bc frequency effects**: As shown in Figure 4, FPS is inversely proportional to the increased CPU workload above 2, and proportional to the increased CPU frequency. Power is proportional to the increased CPU workload only below workload 2, and proportional to the increased CPU frequency. And, EpF is proportional to the increased CPU workload, and almost proportional to the increased CPU frequency.

**Specific Observations**: As shown in Figure 4, using 1800Mhz CPU frequency is better in terms of FPS, power and EpF for all CPU workloads, which means that the default CPU governor is too interactive for CPU dominant workloads, so more conservative policy can be applicable for energy saving.

![Fig. 4: Results of mb-App at different CPU frequencies](image)

3) **mb-TexM**: As shown in Table I, the texture image size was changed for mb-TexM, and we fixed a certain amount of numbers of vertices and triangles; we minimized stresses of other stages. The x-axis is the texture image size, and the y-axis is FPS, Power, and energy per frame (EpF) respectively as shown in Figure 5.

![Fig. 5: Results of mb-TexM](image)
a) **GPU frequency effects**: As shown in Figure 5, FPS is inversely proportional to the increased image size above 1024 (frame time increase by GPU workload increase) and almost similar in GPU frequency change. However, power is proportional to the increased texture image size below 2048, and proportional to the increased GPU frequency below 2048 except 266Mhz (power is proportional to frequency and voltage). And, EpF is proportional to the increased texture image size for all workloads, and proportional to the increased GPU frequency below 2048 except 266Mhz.

Specific Observations: As shown in Figure 5(a), almost similar FPS was observed by GPU frequency changes for all workloads (if we assume that the frame time is composed of scalable time and non-scalable time, we can speculate that non-scalable time could be dominant by increased pipeline stall time or memory access latency increase (i.e., memory bottleneck)). And as shown in Figure 5(b) and (c), lower GPU frequency (except 266Mhz) is better in terms of power and EpF for memory dominant (bottleneck) workloads, and Power and EpF of 266Mhz are extraordinarily higher than other frequencies for overall workloads.

![Fig. 5: Results of mb-TexM at different GPU frequencies](image)

b) **CPU-bc frequency effects**: As shown in Figure 6, FPS is inversely proportional to the increased texture image size above 1024 and proportional to CPU frequency change above 2048. Power is proportional to the texture image size for all workloads and CPU frequency change below 1024. And, EpF is proportional to texture image size and CPU frequency change.

Specific Observations: As shown in Figure 6(b), the power difference between 1800Mhz and 2000Mhz CPU frequency is huge for all workloads.

![Fig. 6: Results of mb-TexM at different CPU frequencies](image)

4) **mb-VerSh and mb-FragSh**: We observe that the experimental results for mb-VerSh and mb-FragSh are similar to mb-VerM except the amount of average power consumption. Therefore, we move those results to Appendix A.
A First Cut: from the results of micro-benchmarks

- For all GPU related micro-benchmarks, 266Mhz GPU frequency is not good in terms of power and EpF, and for all micro-benchmarks.

Before describing the reason, the power and energy inefficiency of 266Mhz GPU frequency was also observed through results of all combinations of micro-benchmarks. This means that this issue is a platform specific issue rather than general graphics workloads issue. Therefore we studied voltage configuration and setting in each GPU frequency. According to our investigation, the highest voltage value was set for this frequency (266Mhz) because unmatched frequency setting between files was observed (i.e., in stead of 266, actually 260 in one header file (asv-exynos5422_evt0.h) was used). After fixing the frequency of the header file, we measured again power consumption of mb-VerM by GPU frequency change, and we couldn’t observe the exception of 266Mhz. Therefore, we will not describe this issue any more in the following sections (but, just record with [GPU 266Mhz issue]).

- Power difference between CPU 1800Mhz (1.8Ghz) and 2000Mhz (2.0Ghz) is huge compared to other frequencies.

This pattern was also observed through results of all combinations of micro-benchmarks. And according to our current investigation, 2.0Ghz is using higher memory frequency which also has higher memory voltage, compared to 1.8Ghz. However, currently more investigation is needed to know exact reasons for this issue. Like the issue of GPU 266Mhz, we just record this issue with [CPU 2.0Ghz issue] in the following sections.
II. RESULTS OF COMBINATIONS OF THE MICRO-BENCHMARKS

In order to design combinations of micro-benchmarks, we choose three micro-benchmarks (mb-App, mb-VerM, mb-TexM). mb-App, mb-VerM, and mb-TexM represent (emulate) CPU, GPU, and Memory (mainly GPU memory) workload respectively. In particular, we choose mb-VerM among mb-VerM, mb-VerSh, and mb-FragSh to represent GPU workload as described in the previous section.

![Fig. 7: Roadmap of Combinations of micro-benchmarks.](image)

The goal of this design is to cover various types of graphics workloads as much as possible in a fixed amount of time. Figure 7 describes the combinations of micro-benchmarks, which is composed of two parts: one is combinations of two micro-benchmarks and the other is combinations of three micro-benchmarks. The reason that we separate these two steps is that we want to observe as specific as possible the effects of each micro-benchmark in combined workloads.

A. Combinations of two micro-benchmarks

The combination of two micro-benchmarks is composed of 1) mb-App med + mb-VerM change, 2) mb-App med + mb-TexM change, 3) mb-App high + mb-VerM change, and 4) mb-App high + mb-TexM change.

1) **mb-App med + mb-VerM change**: As shown in Figure 7, for this combination, after fixing CPU workload with medium workload (level 2), and then we changed the number of vertices like mb-VerM.
a) **GPU frequency effects**: As shown in Figure 8, FPS is inversely proportional to the number of vertices and proportional to the GPU frequency change above 384 workload. Power is proportional to the number of vertices below 256 workload. And, EpF is proportional to the increased number of vertices for all workloads, and inversely proportional to GPU frequency change above 256 workload.

**Specific Observations**: As shown in Figure 8, the graph pattern of this combination is similar to the pattern of 'mb-App' (CPU dominant case) below 256 vertices, and similar to the pattern of 'mb-VerM' (GPU dominant case) above 384 vertices. And as shown in Figure 8(b) and (c), higher GPU frequencies (480Mhz and 543Mhz GPU frequencies) are better in terms of power and EpF than others for below 256 (CPU dominant workloads). [GPU 266Mhz issue].

![Fig. 8: Results of 'mb-App med + mb-VerM change' at different GPU frequencies](image)

b) **CPU-bc frequency effects**: As shown in Figure 9, FPS is proportional to CPU frequency change except 512 workload. Power and EpF are proportional to the number of vertices and CPU frequency change.

**Specific Observations**: According to our observations, this combination is CPU dominant below 256 vertices, and GPU dominant above 384 vertices. As shown in Figure 9(b), the power of the default is similar to the power of 2000Mhz for CPU dominant case, and similar to power of 1600Mhz (reduced from 2000Mhz) in GPU dominant case. Through this observation, we speculate that GPU workload effects are more dominant according to increase of the number of vertices. [CPU 2.0Ghz issue].

![Fig. 9: Results of 'mb-App med + mb-VerM change' at different CPU frequencies](image)

2) **mb-App med + mb-TexM change**: As shown in Figure 7, for this combination, after fixing CPU workload with medium workload (level 2), and then we changed the texture image size like mb-TexM.

![Fig. 7: Results of 'mb-App med + mb-TexM change' at different texture image sizes](image)
a) GPU frequency effects: As shown in Figure 10, FPS is inversely proportional to the increased texture image size above 1024, even though the FPS of 2048 is almost similar to that of 1024. However, there is no significant FPS change by the GPU frequency change for all workloads. Power is proportional to the texture image size except 4096. And, EpF is proportional to the texture image size for all workloads.

Specific Observations: As shown in Figure 10(b) and (c), the graph pattern of this combination is similar to the pattern of 'mb-App' that GPU frequencies of 543Mhz and 480Mhz are better in terms of Power and EpF, and similar to the pattern of 'mb-TexM' that lower GPU frequency is better in terms of Power and EpF. Note that these two patterns can be observed for all workloads. Through this observation, we speculate that CPU workloads and GPU memory workloads can exist independently because one belongs to scalable part and the other belongs to non-scalable part. [GPU 266Mhz issue].

![Fig. 10: Results of 'mb-App med + mb-TexM change' at different GPU frequencies](image)

b) CPU-bc frequency effects: As shown in Figure 11, FPS is inversely proportional to the increased texture image size above 1024 workload, and proportional to the CPU frequency change for all workloads. Power is proportional to the texture image size below 1024, and proportional to the CPU frequency change for all workloads. And, EpF is proportional to the texture image size and the CPU frequency change.

Specific Observations: As shown in Figure 11, the graph pattern of this combination is similar to the pattern of 'mb-App' for all texture image size. [CPU 2.0Ghz issue]

![Fig. 11: Results of 'mb-App med + mb-TexM change' at different CPU frequencies](image)

3) mb-App high + mb-VerM change and mb-App high + mb-TexM change: We observe that the experimental results for 'mb-App high + mb-VerM change' are similar to those of 'mb-App med + mb-VerM change' except pattern shift by workload increase ('mb-App high + mb-TexM' similar to 'mb-App med + mb-TexM'). Therefore, we move those results to Appendix B.
B. Combinations of three micro-benchmarks

The combinations of three micro-benchmarks comprise 5) mb-VerM med + mb-TexM med + mb-App change and 6) mb-VerM low-med + mb-TexM med-high + mb-App as shown in Figure 7.

1) mb-VerM med + mb-TexM med + mb-App change: As shown in Figure 7, for this combination, after fixing mb-VerM medium workload (the number of 256 vertices) and mb-TexM medium workload (the texture image size of 1024), and then we changed CPU workload like mb-App.

   a) GPU frequency effects: As shown in Figure 12, FPS is inversely proportional to the CPU workload above workload 2, and proportional to the GPU frequency change below workload 1. Power is proportional to the CPU workload below workload 1. EpF is proportional to the CPU workload for all workloads. Specific Observations: As shown in Figure 12, the graph pattern of this combination is similar to the pattern of ‘mb-VerM’ (GPU dominant case) below workload 2, and similar to the pattern of ‘mb-App’ (CPU dominant case) above workload 2. In particular, even though mb-VerM medium workload and mb-TexM medium workload were used at the same time, the effects of mb-VerM workload were dominant below workload 2. Therefore, additionally we will measure and analyze the results of the case 6 (i.e, mb-VerM low-med (lower mb-VerM workload) + mb-TexM med-high (higher mb-TexM workload) + mb-App) in the next subsection (Section II-B2). [GPU 266Mhz issue]

Fig. 12: Results of ’mb-VerM med + mb-TexM med + mb-App’ at different GPU frequencies

b) CPU-bc frequency effects: As shown in Figure 13, FPS is inversely proportional to CPU workload and proportional to the CPU frequency change above workload 2. Power is proportional to the CPU workload below workload 1, and proportional to CPU frequency change for all workloads. EpF is proportional to the CPU workload and CPU frequency change for all workloads. Specific Observations: As shown in Figure 13(a), the pattern of ‘mb-App’ (CPU dominant case) is clearly observed above workload 2 according to CPU workload increase. [CPU 2.0Ghz issue].

Fig. 13: Results of ’mb-VerM med + mb-TexM med + mb-App’ at different CPU frequencies
2) mb-VerM low-med + mb-TexM med-high + mb-App: As shown in Figure 7, in order to increase the effects of mb-TexM compared to mb-VerM, after fixing mb-VerM low-medium workload (the number of 128 vertices) and mb-TexM medium-high workload (the texture image size of 2048), we changed CPU workload like mb-App.

a) GPU frequency effects: As shown in Figure 14, FPS is proportional to the increased CPU workload below workload 1, but inversely proportional to the increased CPU workload above workload 2. And there is no significant FPS change by the GPU frequency change. Power is proportional to the CPU workload below workload 1, but is inversely proportional to the increased CPU workload above workload 2. EpF is proportional to the CPU workload for all workloads, and proportional to the GPU frequency change above workload 2. [GPU 266Mhz issue].

Specific Observations: As shown in Figure 14, the graph pattern of this combination is similar to the pattern of 'mb-TexM' (GPU dominant case) below workload 1, and similar to the pattern of 'mb-TexM' affected by 'mb-App' (CPU dominant case) above workload 2. And as shown in Figure 14(b) and (c), using lower GPU frequency is better in terms of Power and EpF (which is the pattern of mb-TexM intensive workload).

Fig. 14: Results of 'mb-VerM low-med + mb-TexM med-high + mb-App' at different GPU frequencies

b) CPU-bc frequency effects: As shown in Figure 15, FPS is proportional to the CPU workload below workload 1, but is inversely proportional to the CPU workload above workload 2 (proportional to the CPU frequency change above workload 2). Power also is proportional to the CPU workload below workload 1, but is inversely proportional to the increased CPU workload above workload 2. EpF is proportional to the increased CPU workload and increased CPU frequency change for all workloads.

Specific Observations: As shown in Figure 15(a), this combination is GPU dominant below workload 1, and CPU dominant above workload 2. And as shown in Figure 15(b) and (c), it is better to use lower CPU frequency for GPU dominant part in terms of Power and EpF. [CPU 2.0Ghz issue].

Fig. 15: Results of 'mb-VerM low-med + mb-TexM med-high + mb-App' at different CPU frequencies
III. SUMMARY AND NEXT PHASE

In this section, we summarize significant observations which could be directly applicable for DVFS design or/and useful guidelines. And then, we describe the reasons of doing this phase (phase 3.1), and the next phase (phase 3.2).

A. Summary of each micro-benchmark

- mb-VerM, mb-VerSh and mb-FragSh have similar graph patterns except the amount of average power consumption and very low memory utilization (below 25%) for all cases.

We speculate that these kinds of micro-benchmarks correspond to GPU workloads that has little memory workloads. And we use mb-VerM as a representative of the three micro-benchmarks based on the observations: 1) FPS, power, and EpF graph patterns of the three micro-benchmarks are almost similar except the amount of average power consumption. 2) Increasing the number of vertices is also correlated with workloads of vertex and fragment shaders (stages after vertex fetcher stage). 3) Average power consumption of mb-VerM for all measured cases was middle among the three micro-benchmarks. Moreover, even though we categorized mb-VerM as GPU memory region in Phase 2, we speculate that mb-VerM workloads do not go through memory bottleneck on Exynos 4422 SoC and that they are very similar to GPU computation related workloads.

- mb-App is mainly related to the scaling effects of CPU-bc frequency compared to those of GPU frequency.

mb-App corresponds to CPU workloads that has little memory workloads. According to our observations, the default CPU governor is too interactive (or ondemand) for CPU dominant workloads. Therefore, more conservative policy can be applicable for energy saving without significant FPS degradation. Moreover, higher GPU frequency and lower CPU frequency compared to the default CPU- and GPU governors’ settings could be applicable for power and energy saving with a little bit lower FPS.

- mb-TexM has no significant FPS change by GPU frequency scaling and high memory utilization (over 60%) for medium and high workloads.

We speculate that mb-TexM is mainly related to GPU memory workloads in addition to GPU workloads. Almost similar FPS was observed by GPU frequency changes for all workloads. If we assume that the frame time is composed of scalable time and non-scalable time, we can speculate that non-scalable time could be dominant by increased pipeline stall time or memory access latency increase (i.e., memory bottleneck). Therefore, lower GPU frequency is better in terms of power and EpF saving without FPS reduction for memory dominant workloads.

- For all GPU related micro-benchmarks, 266Mhz GPU frequency is not good in terms of power and EpF, and for all micro-benchmarks.

Before describing the reason, the power and energy inefficiency of 266Mhz GPU frequency was also observed through results of all combinations of micro-benchmarks. This means that this issue is a platform specific issue rather than general graphics workloads issue. Therefore we studied voltage configuration and setting in each GPU frequency. According to our investigation, the highest voltage value was set for this frequency (266Mhz) because unmatched frequency setting between files was observed (i.e., in stead of 266, actually 260 in one header file (asv-exynos5422_evt0.h) was used). After fixing the frequency of the header file, we measured again power consumption of mb-VerM by GPU frequency change, and we couldn’t observe the exception of 266Mhz.
• Power difference between CPU 1800Mhz (1.8Ghz) and 2000Mhz (2.0Ghz) is huge compared to other frequencies.

This pattern was also observed through results of all combinations of micro-benchmarks. And according to our current investigation, 2.0Ghz is using higher memory frequency which also has higher memory voltage, compared to 1.8Ghz. However, currently more investigation is needed to know exact reasons for this issue.

B. Summary of two micro-benchmarks

• mb-App med + mb-VerM is combination of CPU workloads and GPU workloads with minimized GPU memory workloads, and pattern change from CPU dominant to GPU dominant effects exist. The graph pattern of this combination is similar to the pattern of ‘mb-App’ (CPU dominant case) below medium workload, and similar to the pattern of ‘mb-VerM’ (GPU dominant case) above medium-high workload. Through the observations, we speculate that effects of one prominent workload among combined (mixed) workloads affect dominantly in a scalable part.

• mb-App med + mb-TexM is combination of CPU workloads and GPU memory workloads, and both CPU- and GPU dominant effects exist at the same time. The graph pattern of this combination is similar to the pattern of ‘mb-App’ that GPU frequencies of 543Mhz and 480Mhz are better in terms of Power and EpF, and similar to the pattern of ‘mb-TexM’ that lower GPU frequency is better in terms of Power and EpF. However, note that these two patterns can be observed for all workloads. Through this observation, we speculate that CPU workloads and GPU memory workloads can exist independently because one is scalable part (mainly by CPU) and the other is non-scalable part (mainly by GPU).

C. Summary of three micro-benchmarks

• When many workloads such as mb-VerM + mb-TexM + mb-App are combined, representative CPU workload or/and one of GPU workloads (i.e., mb-VerM or mb-TexM) is/are dominant. For instance, in ‘mb-VerM med + mb-TexM med + mb-App’ case, the graph pattern is similar to the combination of ‘mb-App + mb-VerM’. However, for ‘mb-VerM low-med + mb-TexM med-high + mb-App’ case, the graph pattern is similar to the combination of ‘mb-App + mb-TexM’.

D. For the Next Phase

In this phase (phase 3.1), we described the results of each micro-benchmark and combinations of micro-benchmarks to utilize these results for DVFS policy design. In order to design policy systematically, first of all, opportunities of improvements in energy saving with minimal FPS degradation for different types of mixed workloads should be observed and characterized. Using these general and specific opportunities, more energy-efficient policy can be designed.

And then, in the next phase (phase 3.2), we correlate (emulate) sample graphics applications (e.g., mobile games) to specific combined micro-benchmarks to estimate improvements in power and performance efficiency using analysis of the results of the phase 3.1; outline opportunities for synergistic CPU-GPU memory optimizations and mobile DVFS design; and propose sample integrated CPU-GPU DVFS policies for quality-aware energy savings.
IV. APPENDICES

A. APPENDIX A

Result and Analysis on mbVerSh and mbFragSh

A.1 mb-VerSh
As shown in Table I, the number of executions of vertex shader program was changed for the mb-VerSh with minimized stresses of other stages. The x-axis is the number of executions of vertex shader program, and the y-axis is FPS, Power, and energy per frame (EpF) respectively as shown in Figure 16.

**a) GPU frequency effects:** As shown in Figure 16, FPS is inversely proportional to the number of executions of vertex shader program, and proportional to the increased GPU frequency change. Power is proportional to the increased CPU frequency except 266Mhz. And, EpF is proportional to the increased number of executions of vertex shader program.

**Specific Observations:** As shown in Figure 16(b) and (c), Power and EpF of 266Mhz are extraordinarily higher than other frequencies for overall workloads. And as shown in Figure 16(c), 177Mhz and 266Mhz GPU frequency is not efficient for overall workloads except the minimum workload (level 1).

![Fig. 16: Results of mb-VerSh at different GPU frequencies](image1)

**b) CPU-bc frequency effects:** As shown in Figure 17, FPS is inversely proportional to the number of executions of vertex shader program above 50000 executions, but is not changed by CPU frequency change (because of the GPU dominant workloads). Power is proportional to the increased number of executions of vertex shader program below 70000 executions, and proportional to the increased CPU frequency change. And, EpF is proportional to the proportional to the increased number of executions of vertex shader program, and proportional to the increased CPU frequency change.

**Specific Observations:** As shown in Figure 17(b) and (c), Power and EpF of default CPU governor are always similar with them of 1200Mhz, because this test case has low CPU workload. And as shown in Figure 17(b), the power difference between 1800Mhz and 2000Mhz CPU frequency is huge for all cases.

![Fig. 17: Results of mb-VerSh at different CPU frequencies](image2)
A.2 mb-FragSh
As shown in Table I, the number of executions of fragment shader program was changed for mb-FragSh and we minimized stresses of other stages. The x-axis is the number of executions of fragment shader program, and the y-axis is FPS, Power, and energy per frame (EpF) respectively as shown in Figure 18.

 **a) GPU frequency effects:** As shown in Figure 18, FPS is inversely proportional to the increased number of executions of fragment shader program above 512 executions, and proportional to the increased GPU frequency change above 512 executions. Power is proportional to the increased number of executions of fragment shader program below 512 executions, and proportional to the GPU frequency except 266Mhz. And, EpF is proportional to increased number of executions of fragment shader program.

 **Specific Observations:** As shown in Figure 18(b) and (c), Power and EpF of 266Mhz are extraordinarily higher than other frequencies for overall workloads.

![Fig. 18: Results of mb-FragSh at different GPU frequencies](image)

 **b) CPU-bc frequency effects:** As shown in Figure 19, FPS is inversely proportional to the increased number of executions of fragment shader program above 512 executions. Power is proportional to the increased number of executions of fragment shader program below 768 executions, and proportional to increased CPU frequency change. And, EpF is proportional to the increased number of executions of fragment shader program, and proportional to the increased CPU frequency change.

 **Specific Observations:** As shown in Figure 19(b) and (c), the pattern of graph is similar with Figure 2 (‘CPU frequency effects’ of ’mb-VerM’) and Figure 17 (‘CPU frequency effects’ of ’mb-VerSh’). And as shown in Figure 19(b), the power difference between 1800Mhz and 2000Mhz CPU frequency is huge for all cases.

![Fig. 19: Results of mb-FragSh at different CPU frequencies](image)
B. APPENDIX B

Result and Analysis on ’mb-App high + mb-VerM change’ and ’mb-App high + mb-TexM change’

B.1 mb-App high + mb-VerM change
As shown in Figure 7, after fixing CPU workload in level 4 (the highest level), the number of vertices was changed with minimized stresses of other stages for this combination.

a) GPU frequency effects: As shown in Figure 20, there is no significant FPS change by the number of vertices (0 of the high CPU workload). Power is proportional to the increased number of vertices for all cases, but not proportional to the GPU frequency change. And, 3 is proportional to the increased number of vertices for all cases, but not proportional to the GPU frequency change.

Specific Observations: As shown in Figure 20, the graph pattern of this combination is similar to the case of CPU intensive workloads as shown in Figure 3 except 512 workload (the effect of mb-VerM).

![Graphs of FPS, Power, and EpF for different number of vertices and GPU frequencies.](image)

Fig. 20: Results of ’mb-App high + mb-VerM change’ at different GPU frequencies

b) CPU-bc frequency effects: As shown in Figure 21, FPS is proportional to the increased CPU frequency change, but there is no significant FPS change by the number of vertices (because of the high CPU workloads). Power is proportional to the number of vertices and the increased CPU frequency change. And, EpF is also proportional to the number of vertices and the increased CPU frequency change.

Specific Observations: As shown in Figure 21(a), because of the high CPU workloads, the highest FPS also is very low. And general graph pattern is similar to that of mb-App.

![Graphs of FPS, Power, and EpF for different number of vertices and CPU frequencies.](image)

Fig. 21: Results of ’mb-App high + mb-VerM change’ at different CPU frequencies
**B.2 mb-App high + mb-TexM change**

As shown in Figure 7, after fixing CPU workload in level 4 (the highest level), the texture image size was changed for this combination with minimized stresses of other stages.

**a) GPU frequency effects:** As shown in Figure 20, FPS is inversely proportional to texture image size above 512, but there is no significant FPS change by the GPU frequency change. Power is proportional to texture image size for all cases, but there is no significant power change by the GPU frequency change except 256. Therefore, EpF is proportional to the texture image size, but there is no significant power change by the GPU frequency change except 256.

**Specific Observations:** As shown in Figure 20(b) and (c), using high GPU frequency is better in terms of Power and EpF. (this pattern was observed in mb-App (Figure 3).

---

**b) CPU-bc frequency effects:** As shown in Figure 21, FPS is inversely proportional to the texture image size, and proportional to the increased CPU frequency change. Power is proportional to the texture image size and the increased CPU frequency change. Therefore, EpF also is proportional to the texture image size and the increased CPU frequency change.

**Specific Observations:** As shown in Figure 23, the default governor always use 2000Mhz CPU frequency because of intensive CPU workloads. [CPU 2.0Ghz issue].

---

![Graphs](image)