electronics

Article

Phase-Based Low Power Management Combining CPU and GPU for Android Smartphones

Seung-Ryeol Ohk [1], YongSin Kim [2] and Young-Jin Kim [1,*]

1 Department of Electrical and Computer Engineering, Ajou University, Suwon 16499, Korea
2 Agency for Defense Development, Daejeon 34186, Korea
* Correspondence: youngkim@ajou.ac.kr

Abstract: Smartphones have limited battery capacity, so efficient power management is required for high-performance applications and to increase usage time. In recent years, efficient power management of smartphones has become very important as the demand for power use of smartphones has grown due to deep learning, games, virtual reality, and augmented reality applications. Existing low-power techniques of smartphones focus only on lowering power consumption without considering actual power consumption based on utilization of the central processing unit (CPU) and graphics processing unit (GPU), which are major components of smartphones. In addition, they do not take into consideration the strict use of resources within the component and what instructions are being processed to operate them. In this paper, we propose a low-power technique that manages power by calculating the actual power consumption of smartphones at execution time and classifying the detailed resource operating states of CPUs and GPUs. The proposed technique was implemented by linking the kernel and native app on a Galaxy S7 smartphone equipped with Android. In experiments with 15 workloads, the proposed technique achieves an energy reduction of 18.11% compared to the low-power technique of the interactive governor built into the Galaxy S7 with a small FPS reduction of 3.12%.

Keywords: phase; low power; power estimation; Android; smartphone

Citation: Ohk, S.-R.; Kim, Y.; Kim, Y.-J. Phase-Based Low Power Management Combining CPU and GPU for Android Smartphones. *Electronics* **2022**, *11*, 2480. https://doi.org/10.3390/electronics11162480

Academic Editor: Davide Brunelli

Received: 16 June 2022
Accepted: 8 August 2022
Published: 9 August 2022

1. Introduction

According to [1], as of June 2021, the number of smartphone users worldwide reached 6.4 billion, increasing by 5.3 percent every year. In addition, the use rate of smartphones among mobile handsets has reached 75% [1]. As a result, it is no exaggeration to say that the use of smartphones has now become a daily routine for us. When a user uses a smartphone, battery life is an important factor that affects user satisfaction. However, according to the findings of [2], the biggest dissatisfaction of United Kingdom and United States users is the lack of battery life, and the proportions are 35% and 45%, respectively, for mid-range smartphones and premium smartphones.

There are two ways to increase battery life. The first is to increase the capacity of batteries, and the second is to reduce the power consumption of smartphones. According to a recent trend of battery technology development, the increase in battery capacity is stagnating [3], so the size and weight of batteries generally increase to increase the battery capacity. An increase in the size and weight of the battery will cause an increase in the size and weight of the smartphone, which will cause inconvenience to smartphone users. However, since a large battery capacity is not directly related to a long battery life [4], the power consumption of a smartphone should be reduced to increase the battery life.

In this paper, the component-specific power consumption ratio of the Galaxy S7, a smartphone to be used as a target, is 29%, 56%, 11%, and 4% for the central processing unit (CPU), graphics processing unit (GPU), display, and others, respectively [5]. The sum of the power consumption ratios of the CPU, GPU, and display components is 96%, accounting for most of the power consumption. Therefore, reducing the power consumption of the three components is directly related to the overall power consumption of the smartphone, and in this paper, among other things, we focus on reducing the power consumption of the CPU and GPU.

The power consumption of the CPU and GPU varies according to driving voltage and frequency. Based on the complementary metal-oxide-semiconductor circuit, the power consumption of the CPU and GPU is generally proportional to the frequency and the square of the voltage. Dynamic voltage and frequency scaling (DVFS) is a low-power technology using this point. DVFS is a technique that adjusts voltage and frequency to obtain the optimal trade-off between the power consumption and performance of the processor. The interactive governor, a DVFS governor, is basically included in a smartphone with an Android operating system (OS) based on the Linux kernel and is designed to manage power considering utilization [6]. In addition, in HiCAP, a low-power technique for Android mobile devices, DVFS was designed using CPU utilization and GPU utilization [7].

Low-power techniques using utilization simply consider the degree of CPU and GPU usage. Linux interactive government and HiCAP perform DVFS considering the utilization of the CPU or GPU, but utilization simply informs about the usage, so we do not know what commands are being processed on the CPU or GPU. Knowing which commands the CPU and GPU are currently processing will help in the design more power-efficient DVFS, as the internal hardware resources of the CPU and GPU may vary in how they are processed. In this paper, we propose a low-power technique that monitors the types of commands processed by the processor with a performance-monitoring counter (PMC), divides the case according to the detailed operating states of the CPU and GPU, and works based on the phase. In this paper, phase is the execution state classified according to the usage ratio of various instructions in the processor (CPU and GPU) occupied by the workload. The CPU phase is determined by the cache miss rate and the load and store (LDST) inst rate, which indicates the rate of instructions related to the CPU's memory operation. The GPU phase is determined by the usage ratio of units in the GPU microarchitecture. Detailed and accurate explanations of the CPU/GPU phase are presented in Section 4.

In addition, the above two techniques use a method of lowering the frequency (or voltage) without considering the phase or the real-time power consumption of the CPU and GPU. Of course, since frequency directly affects the processor's power consumption, power can be reduced through the DVFS technique, but if the processor's power consumption is known and power is managed in real time, it would be possible to design DVFS for more accurate and efficient power management.

On the other hand, reducing power consumption through DVFS is possible by reducing the frequency of the processor, but reducing the frequency may reduce the performance of the processor, which may reduce the quality of service (QoS) of the running task. Therefore, it is possible to implement a low-power technique without lowering QoS only when frequency is appropriately controlled. In this paper, frames per second (FPS) is selected as the QoS to confirm the effect of the proposed low-power technique on QoS.

In this paper, we propose a DVFS technique that calculates the actual power consumption of CPU and GPU at execution time using a model, predicts the phase based on PMC measurement, and optimizes QoS and power savings based on this. The technique proposed in this paper specifically divides the operational states of CPU and GPU use, which allows for more delicate and efficient control of the processor, and considering power consumption in runtime, it is considered to achieve more effective power savings than DVFS, which simply sees and controls utilization. The proposed technique was implemented at the Linux kernel, *sysfs* file system, Android framework, and native application level of Android-based smartphones, and extensive experiments with multiple 3D game benchmark applications were fulfilled to verify how much the proposed technique improved in terms of power savings and FPS compared to other techniques.

The remainder of this paper is organized as follows. Section 2 describes the interactive governor that is basically installed in Android smartphones. In addition, the DVFS method is described in which the interactive governor reduces power consumption by changing the voltage and frequency of the CPU and GPU. In addition, existing low-power techniques considering QoS and power savings are introduced for Android mobile devices. Section 3 mentions research motivation and contribution points through consideration of phase-based methods for existing utilization-based methods. Section 4 describes the proposed phase-based CPU and GPU integrated low-power technique. Section 5 implements the proposed low-power technique on smartphones and presents experimental results compared with interactive governors and existing studies. Finally, Section 6 concludes the paper after describing the conclusions and future research.

2. Background and Related Work

2.1. DVFS Techniques on Smartphones

Modern processors are generally designed to be operable for multiple frequencies. The amount of work that the CPU can process changes according to the frequency, and as the CPU frequency increases, the CPU's daily processing speed increases, so the amount of work that can be processed increases. GPU frequency also controls the speed of operation of the GPU.

It is possible to change the utilization of the processor by changing its frequency. Increasing the frequency of the processor decreases the utilization of the processor, whereas decreasing the frequency of the processor increases the utilization of the processor. However, for the above relationship to be established, the task of the processor must be kept constant. Additionally, if the frequency is already at its maximum or minimum, it cannot be raised or lowered further. The frequency governor controls frequency so that the processor maintains proper utilization. Frequency governors provided by Linux-based Android operating systems include performance, powersave, and interactive. Performance sets the processor frequency to the highest, whereas powersave sets the processor frequency to the lowest. Unlike these two governments, which perform simple operations, interactive changes frequencies rapidly with the utilization of the processor. A short speaking, interactive governor sets the target utilization of the processor, increasing the frequency if the current utilization is greater than the target utilization and decreasing the frequency if the target utilization is less than the target utilization.

The interactive governor uses a DVFS technique that adjusts the frequency according to the load of the processor. In Android-based smartphones, frequency and voltage have a fixed number of pairs, and when frequency is changed, voltage is implemented to change accordingly. The frequency is changed according to the set conditions to control the power consumption of the processor. If the frequency of the processor is lowered unconditionally, the power consumption of the smartphone can be absolutely reduced, but if the frequency of the processor is lowered, the performance decreases as well, so the quality of work in progress on the smartphone cannot be guaranteed. However, the interactive governor uses a simple DVFS technique based on utilization, so it does not closely address the trade-off that exists between power consumption reduction and performance improvement.

In recent mobile systems, the DVFS of the CPU and GPU is controlled by a DVFS manager operating independently. On the Samsung Galaxy S Series, CPU DVFS administrators operate as part of heterogeneous multi-processing (HMP). HMP is a model that allows all physical cores to operate simultaneously. It is the most powerful usage model of the LITTLE architecture. HMP is a standard load scheduler, with high-priority threads assigned to the big core and low-priority threads assigned to the little core. However, the HMP does not predict the energy model (EM) of the CPU.

Meanwhile, in the mobile system, the GPU DVFS manager determines the next frequency based on the previous usage rate. GPUs generally have two operating states: Active and Sleep. Active is the GPU executing the command and Sleep is idle. Like the CPU, the utilization rate is calculated as the ratio of the elapsed time of the previous calculation to the time the GPU remains active. DVFS governor algorithms vary from manufacturer to manufacturer, but in general, if the usage rate is greater than the rising threshold, the frequency increases and decreases if it is lower than the falling threshold.

2.2. Related Work

Low-power studies using DVFS for conventional Android smartphones have been conducted for each component of CPUs, GPUs, and displays that account for a large percentage of power in smartphones [8–12], and most smartphone low-power studies show the use of FPS as QoS to express performance quality [7–13]. Unlike the studies conducted for each component, CoCAP [13] and HiCAP [7] are low-power studies that integrate and manage CPUs and GPUs for mobile devices.

HiCAP is a study that expanded and improved CoCAP and is a low-power technique study that considers utilization, which is the usage of CPUs and GPUs. HiCAP designs CPU and GPU integration government and manages CPU and GPU frequencies together, not independently. First, in this technique, the normalized cost considering the CPU and GPU frequency is calculated as in Equation (1).

$$Normalized\ Cost = \frac{Curr_{Utilization} Curr_{Frequency}}{Max_{Utilization} Max_{Frequency}} \tag{1}$$

Normalized cost is a value for classifying workloads and is classified into four categories, CPU/GPU-bound, CPU-bound, GPU-bound, and CPU/GPU-idle, depending on the normalized cost value of each CPU and GPU. However, since the processor processes various tasks while the workload is being performed, various operating states are shown. Just because the processor utilization remains constant does not guarantee that it is continuing to process the same task. Workload classification using HiCAP's normalized cost is insufficient to accurately determine the operation of CPUs and GPUs dynamically.

HiCAP does not have an algorithm to directly select a frequency and uses an existing frequency governor, but uses a method to control the max frequency. The max frequency is changed according to the cost in real time using the cost-specific max frequency Look-Up-Table (LUT) prepared by CoCAP in another study by the same author. HiCAP changes the max frequency according to workload classified according to normalized cost. Workloads with a high proportion of CPU use control a CPU frequency, and workloads with a high proportion of GPU use control a GPU frequency.

3. Motivational Studies

In this paper, phases are divided according to the use of internal resources (for phase expansion, refer to Section 4). Next, it was confirmed that the result of measuring the change in power consumption and FPS by phase showed a different pattern in the change rate of power consumption and FPS depending on the phase. Tables 1 and 2 are tables showing power consumption and FPS change rates according to CPU and GPU phase, respectively. In the case of the CPU phase, the power consumption changed a lot for phases 1, 2, and 3, and the power consumption changed little for phases 4 and 5. The FPS changed a lot during CPU phase 1, and the remaining CPU phase changed small. The higher the CPU phase, the smaller the memory access ratio, and accordingly, phase 1, which has the largest proportion of CPU operations, has a large power consumption and FPS change rate. In the case of GPU phase, there was little change in FPS when GPU phase was 1 and 4, and the change in FPS was large when GPU phase was 2 and 3. This is the result of GPU phases 2 and 3 being phase related to the shader core that renders graphics and tilers that combine frames, respectively.

As a result, if the DVFS governor of the CPU/GPU considers the importance of frequency scaling according to the CPU/GPU phase, it can be expected to bring about an efficient change in power consumption and FPS. For example, when the CPU phase is 4 or 5, changing the CPU frequency does not significantly change the CPU power consumption, so high performance can be achieved by increasing the CPU frequency. Similarly, in the case of GPUs, changing the GPU frequency when the GPU phase is 1 or 4 does not significantly change the FPS, so the GPU frequency can be lowered to benefit from low GPU power consumption. That is, depending on the workload being executed, the phase of each of the various CPUs and GPUs may be viewed, and the DVFS corresponding thereto may be applied.

Table 1. Change rate of power consumption and FPS according to CPU phase when the CPU frequency is changed.

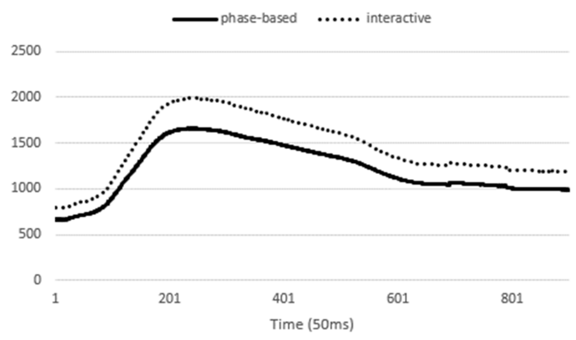
CPU Phase	Power Consumption (%)	FPS (%)
1	−15~22	−4~7
2	−5~5	−2~2
3	−4~5	−1~1
4	−1~1	−1~1
5	−1~1	−1~1

Table 2. Change rate of power consumption and FPS according to GPU phase when the GPU frequency is changed.

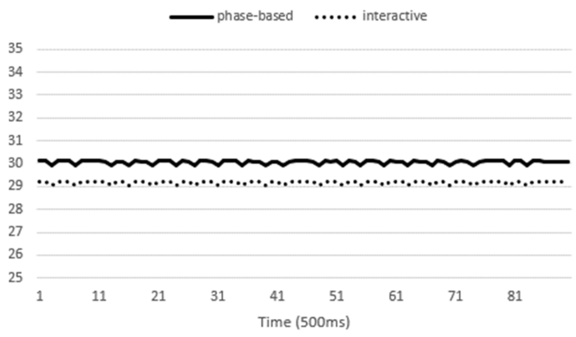
GPU Phase	Power Consumption (%)	FPS (%)
1	32~−14	0~−1
2	16~−20	2~−8
3	16~−16	2~−9
4	16~−15	1~−1

Using such observations, we compared the performance of an interactive governor on the Galaxy S7 that uses utilization-based DVFS techniques with a simple DVFS algorithm based on CPU/GPU phase. Figure 1 is a graph comparing the power consumption and FPS of the original interactive governor based on the utilization- and phase-based interactive governor. Tables 1 and 2 show the change rate of power consumption and FPS according to phase when the frequency of CPU and GPU is changed. CPU and GPU frequencies were changed from 520 MHz to 1768 MHz and from 546 MHz to 650 MHz, respectively. Additionally, the CPU and GPU frequency standards for calculating the change rate of power consumption and FPS are 1144 MHz and 600 MHz, respectively. Based on Tables 1 and 2, the phase-based governor makes the interactive governor raise the CPU frequency by one level when the CPU phase is 4 or 5 and lower the GPU frequency by one level when the GPU phase is 1 or 4. As a result, as shown in Figure 1, the phase-based power consumption is lower and the FPS is higher than that of the interactive, showing more effective DVFS governor performance.

On the other hand, we investigated the effect on power saving and FPS according to pattern classification by phase over normalized cost used in HiCAP. To this end, a method of calculating and using HiCAP's CPU/GPU normalized cost instead of phase in the phase-based interactive governor used earlier was applied. The normalized cost was calculated by Equation (1), and 15 workloads were classified into 4 categories according to the boundness as shown in Figure 2 using the normalized cost. For these 15 workloads, the CPU/GPU frequency was changed to measure the power consumption and FPS for the workload, creating a table of power consumption and FPS change rates. As shown in Table 3, the experimental results show that when the workload is classified based on phase rather than normalized cost, the overall energy savings and small FPS reduction of 2% are based on the large energy savings seen in the GPU. In other words, it can be seen that while reducing the overall power savings of smartphones by 8.9%, the effect on performance is not significant.



(**a**)



(**b**)

Figure 1. Comparisons of (**a**) power consumption and (**b**) FPS for the original interactive governor and phase-based interactive governor.

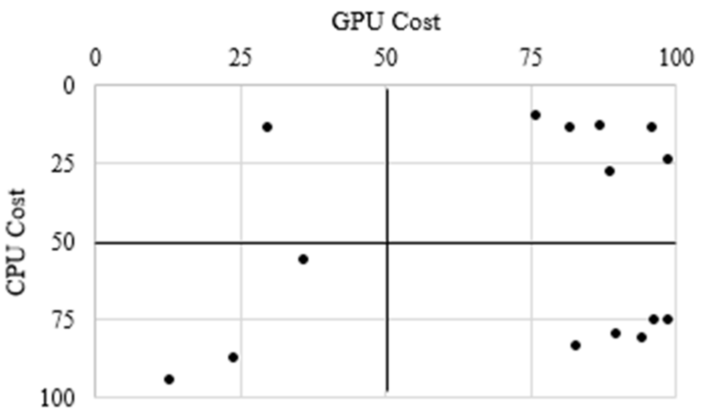


Figure 2. Workloads classified by normalized cost in HiCAP.

Table 3. Comparisons of power saving and FPS of interactive governors classifying workloads by phase over normalized cost.

	Energy					**FPS**
	CPU Little	**CPU Big**	**GPU**	**Display**	**Total**	
Saving Rate	0%	2.2%	14.4%	0%	8.9%	−2%

This paper proposes a new phase-based workload classification technique to improve the classification of existing utilization-based or utilization-like workload patterns for Android mobile devices based on these experimental results. Furthermore, we intend to develop a DVFS technique that is more power- and performance-efficient than the DVFS technique of the existing interactive governor by performing the workload classification delicately and accurately according to the internal resource use of the CPU/GPU. The main contributions of this paper are summarized as follows.

- The proposed phase prediction and management technique makes the workload classification more accurate and provides the basis for system-specific holistic low-power techniques by simultaneous and in-depth consideration of CPU and GPU resource usage when performing workloads.
- By integrating CPU/GPU based on phase, it is possible to design a more power- and performance-efficient DVFS technique than the existing DVFS technique of Linux interactive governor by delicately and accurately performing workload classification according to the internal resource state.
- The proposed phase-based DVFS technique is very practical because it is easy to implement the technique at the governor level in the operating system for mobile devices equipped with Android platforms, including Android smartphones, and it is highly scalable.

4. Phase-Based CPU and GPU Low Power Approach

4.1. Phase Classification

In this paper, "phase" refers to the detailed resource operation states of the CPU and GPU. CPU phase is classified by how often the CPU uses memory and by the memory devices that the CPU mainly uses. Figure 3 shows the phase of the CPU classified into five categories. Two determination criteria are used to decide the phase in Figure 3. First, it is a cache miss ratio or its corresponding alternative value. If the cache miss ratio is high, the CPU uses more dynamic random access memory (DRAM) than the cache in the time interval, so the CPU phase at that time is determined as a DRAM-related phase. Conversely, if the cache miss ratio is low, the CPU uses more cache than DRAM in the corresponding time interval, so the CPU phase at that time is determined as a phase related to the cache. Second, it is an LDST instruction ratio or an alternative value corresponding thereto. Since the LDST instruction ratio represents the ratio of instructions related to the memory operation of the CPU among all instructions, the phase of the CPU is determined by the memory system-related phase (DRAM, cache) or CPU phase as the LDST instruction ratio increases and decreases.

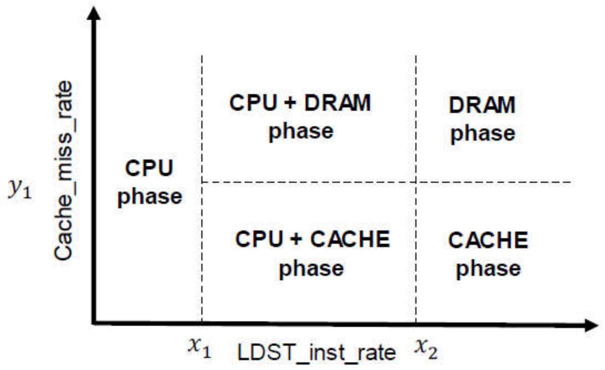


Figure 3. Visualization of CPU phase classification [5].

In addition, the utilization value of each detailed resource, which is a unit, is used as a reference value used to determine the phase of the GPU. Table 4 shows a method of calculating the utilization of each unit of a GPU. These four values are used as a phase determination criterion used in the phase determination method. Meanwhile, the utilization value of the memory unit is different in units of numbers from other utilization values. Therefore, all values are used after a standardization process in which the average is subtracted and divided into standard deviations. In this way, a method of determining a phase using the standardized utilization value of the unit is a maximum-based method. In the present method, a phase corresponding to a unit having a maximum value among four standardized utilization values is selected as a phase of the GPU. Therefore, the phase of the GPU is determined in the same manner as in Equation (2). In Equation (2), FFU means a fixed function unit.

$$Phase_{GPU} = max(util_{FFU},\ util_{Tiler},\ util_{Tripipe},\ util_{Memory}) \tag{2}$$

Table 4. GPU phase decision criteria.

Criteria (Unit Utilization)	Calculation of Unit Utilization
FFU Utilization	(FRAG_ACTIVE + COMPUTE_ACTIV − TRIPIPE_ACTIVE)/GPU_ACTIVE
Tiler Utilization	TI_ACTIVE/GPU_ACTIVE
Tri-pipe Utilization	TRIPIPE_ACTIVE/GPU_ACTIVE
Memory Utilization	L2_ANY_LOOKUP/GPU_ACTIVE

4.2. Data Collection for Proposed Method

Unlike the existing smartphone low-power techniques, this paper proposes a technique that obtains power savings by considering the power consumption of smartphones in execution time. For execution time consumption prediction, the proposed low-power technique uses a power model [5] that predicts CPU and GPU power consumption in real time and a display power model [14] that predicts power consumption using display pixel size. The CPU and GPU power models quickly calculate power consumption within 50 ms through calculations using factors created through reinforcement learning through genetic algorithms. The smartphone CPU and GPU power model is a phase-based power model that divides detailed operating states of the CPU and GPU into several cases and predicts power consumption with an independent power model according to the phase. The display power model may quickly calculate display power by using a LUT. Both power models repeatedly used Monsoon Power Monitor [15] for modeling.

The CPU and GPU power models were designed for the Samsung Exynos M1 and ARM Cortex A-53 CPU and the Mali T880 GPU, which have four little and four big cores and eight cores, respectively. Additionally, the CPU power model used one little core and one big core in the CPU. The GPU power model used whole cores in the GPU. The power consumption of each of the eight CPU cores may be calculated by using the point that the power consumption of each CPU core may be independently calculated.

The power model is also designed to fix CPU big core and GPU frequency at 2184 MHz and 600 MHz, respectively. The frequency is frequently changed because the low-power technique targeted in this paper reduces the power consumption of smartphones by changing the frequency to the DVFS technique. Therefore, a process of correcting the predicted power consumption of the power model according to the changing frequency is required. Equation (3) is used for the predicted power consumption correction according to the frequency change.

$$P = aCV^2f \tag{3}$$

As shown in [16] and [17], the DVFS technique used in this paper is one of the widely known software-level low-power techniques. In Equation (3), P is the power consumption of the CPU, a is the switching activity, C is the capacitance, V is the voltage applied to the CPU, and f is the frequency applied to the CPU. Because the proposed DVFS method uses the average of power consumption calculated several times for the same workload and the same workload performs a fixed task in the processor, a can be considered as a constant value. Additionally, C can be regarded as a constant value because the hardware CPU and GPU are fixed. These are because circuits and applications are fixed in the aspect of the DVFS technique, and the power is related to the square of voltage multiplied by frequency mainly. Therefore, if the voltage and frequency set in the power model are V_{old} and f_{old}, respectively, and the changed voltage and frequency are V_n and f_n, respectively, the power consumption P_{old} predicted by the power model is calculated using Equation (4).

$$P_n = P_{old}\frac{V_n f_n}{V_{old} f_{old}} \tag{4}$$

The power consumption of the GPU is also corrected in the same method as in the CPU power consumption correction Equation (4). CPU and GPU have 21 and 5 variable frequencies, respectively, and the frequency is paired with a voltage determined for each frequency. Therefore, whenever calibrated power consumptions of the CPU and GPU were calculated, the power consumption of each processor was calculated using the LUT without using Equation (4). The power consumption correction of these CPUs and GPUs was used in all power consumption calculations in the low-power technique proposed in this paper.

In addition to power consumption, we collect various data from CPUs and GPUs. The utilization of the CPU and the FPS of the smartphone display are collected at 50 ms intervals. It was observed that the reason for collecting at intervals of 50 ms was that the CPU/GPU's consumption power prediction and data collection process had overhead, so the calculation results were not guaranteed at intervals below 50 ms. Therefore, a minimum period of 50 ms, which can show real-time power consumption while ensuring the accuracy of data such as CPU/GPU predictive power consumption and utilization and FPS, was set as a measurement and collection interval.

The processor utilization used the load value calculated by the interactive governor of the Linux kernel. It is the same calculation as utilization, with the total execution time of the processor minus the time spent in the idle state.

The FPS of the display was calculated using the Android SurfaceFlinger. SurfaceFlinger is responsible for allowing smartphones to synthesize and output screens created by users' processes or applications to be displayed on the display screen. The SurfaceFlinger manages colors, screen locations, and display order, and works with the frame buffer so that final images created with the frame buffer driver present in the Linux kernel can be output to the screen via the frame buffer driver. The HWComposer in Figure 4 counts the number of frames generated in one variable each time the final image synthesis is completed. By dividing the counted number of frames by the time taken, the FPS may be calculated.

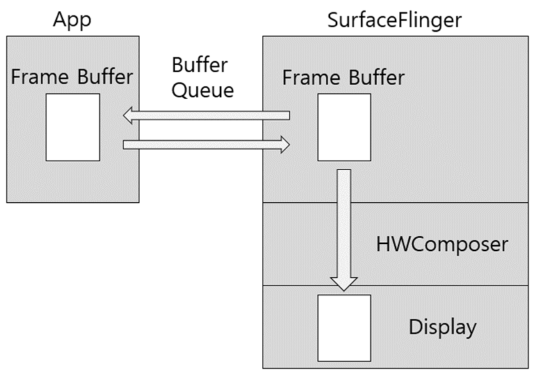


Figure 4. Process of composing and displaying an image by SurfaceFlinger.

4.3. Proposed Method

Figure 5 shows the overall process of the proposed low power management. Our proposed DVFS method proceeds as follows. First, check the CPU utilization to see if the CPU frequency and GPU frequency are changing. Second, check the current phase of the CPU and GPU and predict the next phase. Third, predict the power consumption of the next phase according to the predicted phase. Finally, select the optimal frequency of the CPU and the GPU according to the predicted power consumption. Finally, repeat the above process until the workload is over. Here, the workload is a mobile game application.

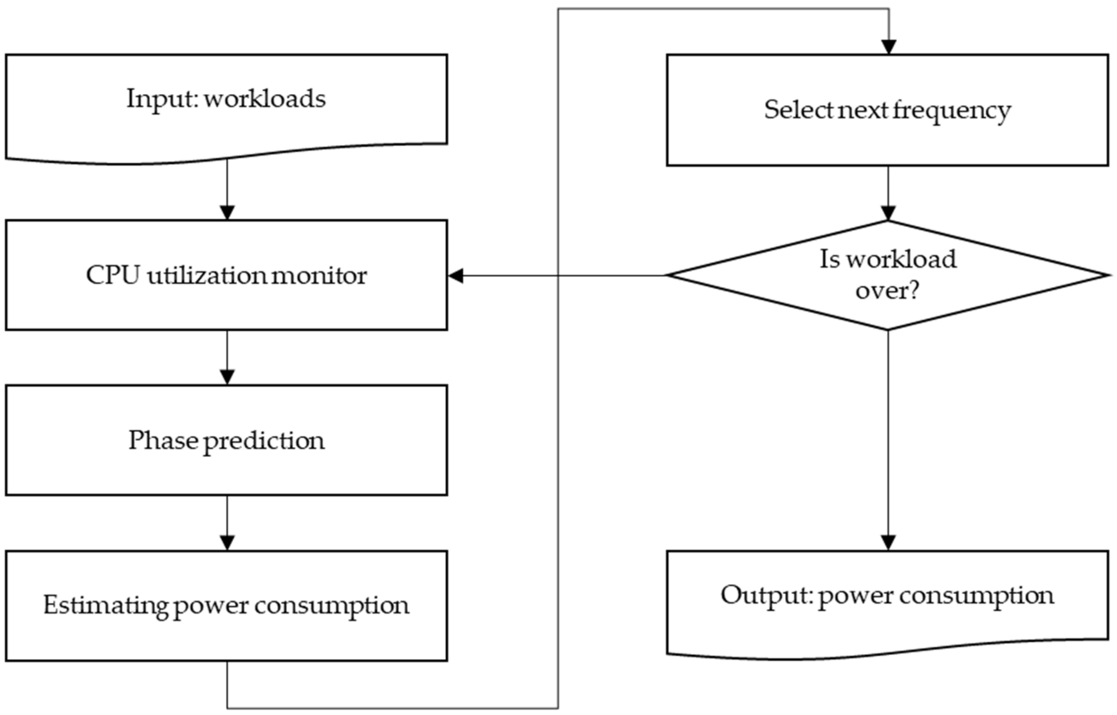


Figure 5. Overall flow of the proposed DVFS method.

Figure 6 shows the part that determines whether to raise, fix, or lower frequency according to CPU utilization, which is the first step of the proposed low-power technique. That is, the frequency of the processor should be changed according to the CPU utilization. There are many cases where the CPU and GPU of a mobile device used in this paper are 21 and 6, respectively, and because of running a workload, the frequency in all cases is not used. Table 5 shows the share by frequency of CPU big core confirmed by running 15 workloads on the basic CPU governor of the Galaxy S7. Of the 21 frequencies, only 5 frequencies were used, not 1664 MHz or higher. In addition, Table 6 is the share of each GPU frequency confirmed by running 15 workloads on the basic GPU governor of the Galaxy S7. As a result of the measurement, it was confirmed that only 650 MHz, 600 MHz, and 546 MHz were used among the six GPU frequencies. Therefore, the proposed low-power technique uses CPU big core frequency between 1560 MHz and 520 MHz, and GPU frequency between 650 MHz, 600 MHz, and 546 MHz.

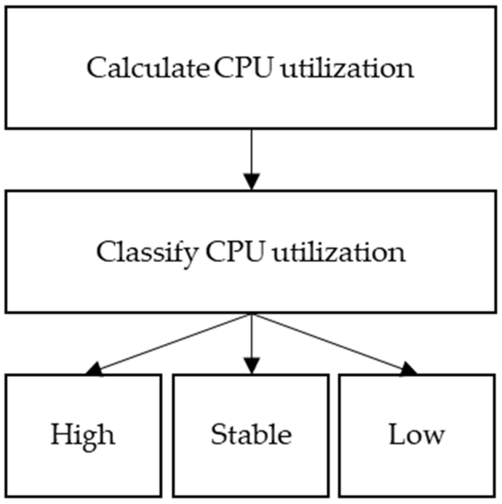


Figure 6. The process of CPU utilization monitoring.

Table 5. CPU big core frequency share.

CPU	Big Core Frequency (MHz)	Occupancy Time
1	1560	70%
2	1456	11%
3	1352	8%
4	1248	6%
5	1144	5%

Table 6. GPU frequency share.

GPU	Frequency (MHz)	Occupancy Time
1	650	70%
2	600	11%
3	546	8%

Now, when we start the low-power technique using only the given frequency, we first check the CPU utilization of the smartphone. If the CPU utilization is 86% or more, we increase the CPU frequency by one step. If the CPU utilization is high, it is lowered by increasing the frequency because CPU throughput is high. When the CPU utilization reaches 100%, it increases the CPU frequency because it can cause tasks that CPU cannot handle, and performance can be degraded. In addition, the optimal threshold for the CPU utilization is found so that the interactive governor, the default CPU frequency governor, increases the CPU frequency when the CPU utilization is over 86%. When the CPU frequency is more than 76% and less than 86%, it maintains the CPU frequency. This is to prevent a rapid repetition of the CPU frequency changes in the saturation case by creating a case where the CPU is fully used, and the CPU frequency is maintained.

If the CPU utilization is 76% or less, the CPU frequency is selected using the power consumption and FPS variation according to the CPU/GPU frequencies for each phase. The next phase should be predicted first. The process of predicting the next phase of the processor is as follows. First, we load a record of the last 7 phases of the processor. Each of these phases has a weight; the more recent, the greater the weight. For example, the phase from the nearest past has a weight of 7, and the subsequent phases have weights of 6, 5, 4, 3, 2, and 1, respectively, in order of closest to the present. Then, the same phases' weights are added and we select the phase with the highest weight as the prediction phase.

The aforementioned two CPU utilization thresholds are set according to the interactive governor [6], which changes CPU frequency according to CPU utilization. The interactive governor set the default values of the two thresholds to 90% and 80%. Through the experiment, we found the optimal CPU utilization thresholds with better performance than the interactive governor's default values. Figure 7 shows the performance score of the proposed low-power technique when experimenting with changing the upper CPU utilization threshold from 90% to 70%. The lower CPU utilization threshold is 10% below the upper one. The performance score is the normalized power consumption reduction rate for the normalized FPS reduction rate. Therefore, the performance score increases when the normalized power consumption reduction rate is high and the normalized FPS reduction rate is low. Additionally, the higher performance score, the better the upper CPU utilization threshold. Therefore, when the upper CPU utilization threshold is 86%, the proposed low-power technique showed the best performance.

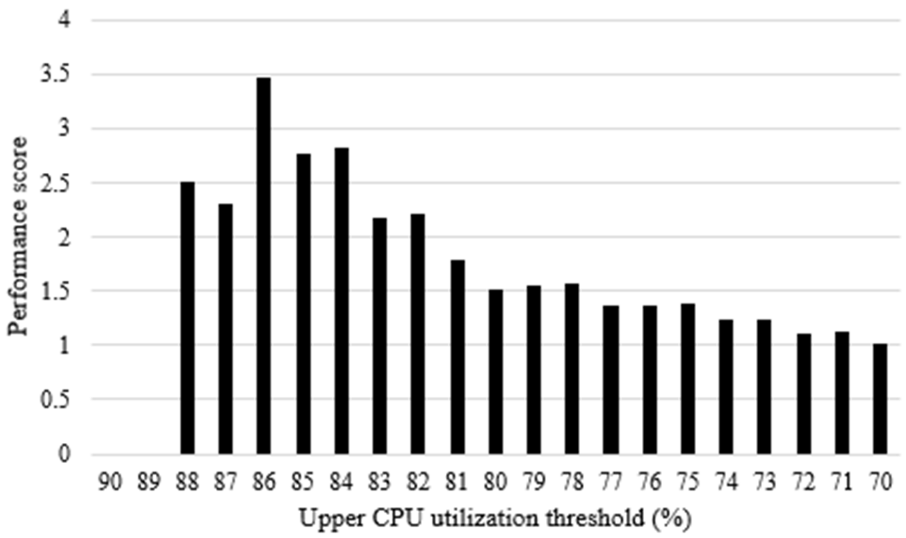
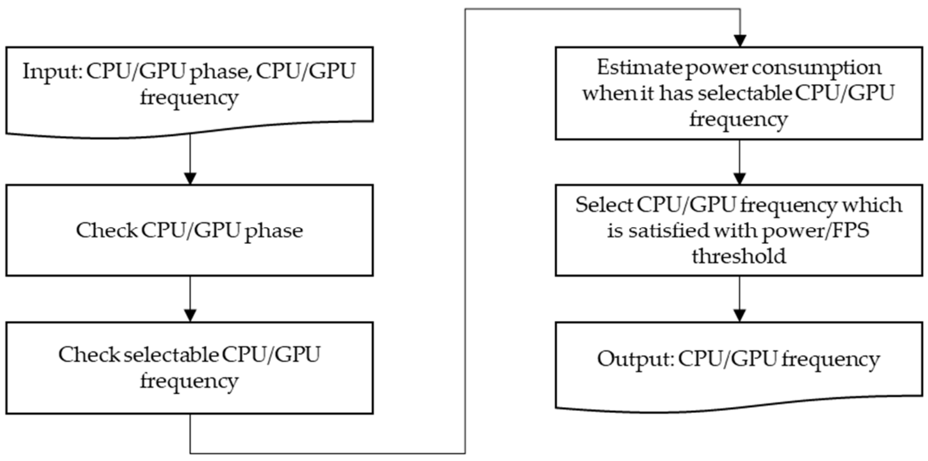


Figure 7. Performance score according to each upper CPU utilization threshold.

If the phase of the next section is predicted, the power consumption of the current section is now calculated. Only the two power consumptions of the CPU big core and GPU are considered. Since the frequency of the CPU little core is not controlled and the display power is not affected by the low-power technique proposed in this paper, the CPU little core and the power of the display are calculated.

If the power consumption in the current section was calculated and the phase in the next section was predicted, the power consumption according to CPU/GPU frequency should now be predicted in the predicted phase. Figure 8 shows the flow of this process. Before that, one should check whether the CPU phase and GPU phase are phases with large power consumption and FPS changes according to the CPU/GPU frequency mentioned above. Here, if the CPU phase or GPU phase is a phase with a small amount of power consumption and FPS change, the frequency of the component is selected as the lowest frequency, and the frequency selected in the next process is ignored. CPU frequency and GPU frequency are 520 MHz and 546 MHz, respectively, the lowest frequencies.



Input: CPU/GPU phase, CPU/GPU frequency
Check CPU/GPU phase
Check selectable CPU/GPU frequency
Estimate power consumption when it has selectable CPU/GPU frequency
Select CPU/GPU frequency which is satisfied with power/FPS threshold
Output: CPU/GPU frequency

Figure 8. The process of selection of next frequency.

If both CPU and GPU frequencies are selected, the process finishes selecting frequency, but otherwise proceeding to the next step. It goes to the section corresponding to the predicted phase in the power consumption and FPS variation table. Next, it finds the location corresponding to the current CPU/GPU frequencies and explores the eight nearest CPUs/GPU frequencies around the location. First, since the current CPU/GPU frequencies may not be the reference frequencies, the process increases or decreases to the same value as the eight surrounding values so that the current frequency consumption power and FPS change amount are 0%. The reference CPU frequency is 1144 MHz, which is the middle value among the CPU frequency range, and the reference GPU frequency is 600 MHz, the middle value among the GPU frequency range.

In addition, the current power consumption is multiplied by the amount of change in the surrounding eight, and the predicted power consumption change is calculated when the CPU/GPU frequencies are changed. For eight variable frequencies, the process adds the predicted power consumption of the CPU and GPU, and then selects the frequency with a little FPS change rate within 5% and the lowest predicted power consumption sum as the frequency of the next interval. The reason why the maximal FPS degradation is set at 5% is that, for workloads over 15 fps, small frame rate differences are not considered important by humans according to [18]. Algorithm 1 shows an algorithm for finding a frequency that minimizes power consumption. We select a frequency by repeating the above process every section until the workload is over.

Algorithm 1 Selecting CPU/GPU Frequency

Input: CPU/GPU Frequency, CPU/GPU Phase
Output: Selected CPU/GPU frequency
P_{CPU}: Predicted CPU phase
P_{GPU}: Predicted GPU phase
F_{CPU}: Current CPU frequency
F_{GPU}: Current GPU frequency
F + i: i th frequency from F in frequency table
PP(p1, p2, f1, f2): Predicted power consumption with CPU/GPU phase p1, p2 and CPU/GPU frequency f1, f2
F'_{CPU}: next CPU frequency
F'_{GPU}: next GPU frequency

```
1 For (i = −1; i <= 1; i++)
2 For (j = −1; j <= 1; j++)
3 Max = 0
4 if (PP($P_{CPU}$, $P_{GPU}$, $F_{CPU}$ + i, $F_{GPU}$ + j < Min)
5 Min = PP($P_{CPU}$, $P_{GPU}$, $F_{CPU}$ + i, $F_{GPU}$ + j)
6 $F'_{CPU} = F_{CPU}$ + i
7 $F'_{GPU} = F_{GPU}$ + j
8 Return $F'_{CPU}$, $F'_{GPU}$
```

As a result, the proposed low-power technique uses algorithms that consider the predicted power consumption when the frequency is changed according to the phase of the CPU and GPU of the Android smartphone at 50 ms intervals to reduce the device's power consumption.

5. Experimental Environment and Results

5.1. Experimental Environment

In this section, CPU- and GPU-integrated low-power techniques are verified through experiments. Specifically, the low power consumption of the workload was calculated by implementing a low power technique proposed for the Linux kernel-based Android-based smartphone, and the basic state without implementing the proposed low power technique and the smartphone consumption energy when running the workload were compared. In addition, smartphone consumption energy was compared when applied to the low-power technique that proposes HiCAP's CPU/GPU cost, and the remaining battery time of the smartphone was calculated to verify the smartphone consumption energy.

The overall experimental environment is shown in Figure 9. After collecting the data required for the low-power technique that runs the workload on the smartphone and proposes it at regular intervals, CPU/GPU frequency can be selected through the low-power technique that suggests it with the collected data. In addition, the energy consumption according to the proposed low-power technique was analyzed and compared with other techniques.

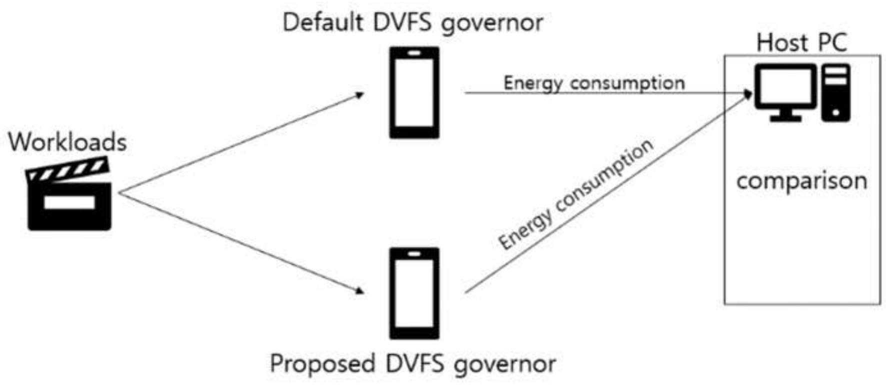


Figure 9. Experimental environment.

The smartphone used in the experiment was a Samsung Galaxy S7. This smartphone uses an Android operating system based on the Linux kernel. The processor uses Exynos 8 Octa (8890) system-on-chip. The processor is equipped with Samsung Exynos M1 MP4 2.29 GHz + ARM Cortex-A53 MP4 1.59 GHz as CPU and ARM Mali-T880 MP12 650 MHz as GPU. The CPU is a big cluster of Samsung Exynos M1 in quad-core configuration and ARM Cortex-A53 in quad-core configuration to form a little cluster in ARM big. It is an octa-core CPU that uses LITTLE solutions.

A total of 15 workloads were used in the experiment, which was installed on the Samsung Galaxy S7 smartphone in the Google Play Store. The installed workloads are 3DMark [19], Madagascar 3D benchmark [20], Kassja benchmark [21], Renderscript benchmark [22], Nena benchmark [23], V1 GPU benchmark [24], Seascape benchmark [25], Basemark [26], 3D benchmark [27], Relative benchmark [28], and L-bench [29]. The 3DMark workload treated the section as five different workloads. In this way, it was used to verify the performance of the low-power technique, which proposes a total of 15 workloads. Figure 10 is a system configuration diagram of the proposed low-power technique. It shows the system in which power consumption, FPS, and various data are calculated and used.

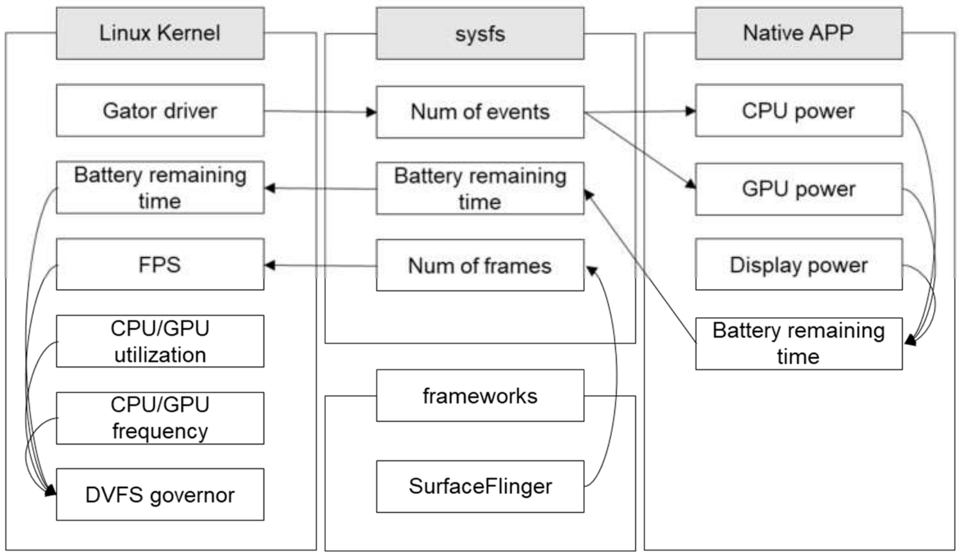


Figure 10. Diagram of the proposed system configuration diagram on the Android platform.

The CPU and GPU power consumption calculation starts with a gator driver [30] in the Linux kernel. The gator driver uses the PMC to measure the number of events performed on the CPU and GPU. There is an operation that multiplies the number of selected events and factors learned using genetic algorithms, and the real and log operations make it difficult for Linux kernels to calculate only integers. As such, we made a native app and calculated CPU and GPU power consumption and sent it to the kernel.

The display power consumption is calculated using the display power model [14]. It is an operation that uses LUT to multiply both the display pixel size and the selected constant, but it cannot be calculated in the Linux kernel for the same reason as CPU and GPU power consumption, so it is calculated using the native app and sent to the kernel. In the SurfaceFlinger, which creates a display frame, the FPS counts frames by one for each frame completed, calculating the frames counted over a unit time (1 s), and sending them to the kernel. Power consumption and FPS are sent via *sysfs* when sent to the kernel. The *sysfs* is a virtual file system provided by the Linux kernel that can send information to the kernel driver in the user space.

In the case of CPU utilization, the CPU load used by the interactive governor provided by Android was used. The CPU load is the time remaining after subtracting the time entering the idle state from the execution time for a constant execution time of the CPU. Therefore, it has a value between 0% and 100%.

The Linux kernel used the EXPORT_SYMBOL command to share data across multiple drivers. All variables and functions were declared so that data could be used within other drivers. The central DVFS governor was created using the CPU interactive governor.

The frequency update period was set to 50 ms as described above. In order to synchronize with the native app, a variable that can recognize the start of the governor cycle was created so that the native app could monitor it. The power calculation speed of the Native App was less than 50 ms, so data on power could be sent no later than the governor's period. All frequency selection processes are performed in the CPU interactive governor, and the selected CPU frequency can be set to the CPU using the function implemented in the CPU frequency driver, and the selected GPU frequency can also be set using the GPU frequency set function implemented by the GPU driver.

To implement the proposed technique, we added a gator driver to enable PMC control at the level of device drivers in Lineage OS [31], which is an open-source operating system based on the Android mobile platform. In addition, the proposed DVFS was implemented by adding a low-power algorithm governor based on the interactive governor. Additionally, the power calculation that requires real part operation was performed using a native app. All these processes can be implemented only by adding source code to the Android operating system and implementing a native app.

5.2. Experimental Results

The proposed low-power technique was compared with the default frequency governor of the Samsung Galaxy S7 to confirm the improved performance of the proposed low-power technique. In addition, the power management advantages were confirmed through phase classification of the proposed low power technique by comparing HiCAP, a related study, with the proposed low-power technique. Finally, the power consumption reduction rate of the proposed low-power technique was verified using the remaining time of the smartphone battery.

The Samsung Galaxy S7 smartphone used in the experiment has an Android operating system based on the Linux kernel. The Linux kernel controls the CPU frequency governor and GPU frequency governor, and the default frequency governor of the Samsung Galaxy S7 smartphone is the CPU interactive governor and GPU, respectively. In this paper, we compared the consumption energy of the proposed CPU- and GPU-integrated low-power techniques with the default frequency governor of the Samsung Galaxy S7 smartphone. To this end, the default frequency governor and the proposed low-power technique were carried out with all eight CPU cores online. For each technique, the same workload was executed and the power consumption of the smartphone from the start to the end was collected. Power consumption is collected in four types: CPU little core, CPU big core, GPU, and display.

In order to calculate the energy consumed to execute one workload as power consumption and time consumption, the energy consumption was calculated using Equation (5).

$$E = \sum P_n t_n \tag{5}$$

In Equation (5), E is the energy consumed when running one workload, P_n is the power consumed at the time interval in which the power model calculates the power consumed, and t_n is the time interval.

In this paper, we compared the energy consumption when using the default frequency governor for 15 workloads with the Samsung Galaxy S7 smartphone and when using the proposed low-power technique. The use of the low-power technique proposed for all workloads used in the experiment showed lower energy consumption than the use of the default frequency governor. The comparable energy consumption is the total energy consumption of the smartphone when the workload is run once. Equation (6) shows that the total energy consumption E_{total} is the sum of all of the energy consumption for the four components calculated using the power model described above: $E_{CPUlittle}$, E_{CPUbig}, E_{GPU}, and $E_{Display}$.

$$E_{total} = E_{CPUlittle} + E_{CPUbig} + E_{GPU} + E_{Display} \tag{6}$$

When comparing the total energy consumption for each workload of the two techniques, the total energy consumption of the proposed low-power technique was less than the default frequency governor for all workloads, so the total energy consumption rate of the proposed technique was calculated through Equation (7).

$$D(\%) = \left| \frac{E_{totalproposed} - E_{totaldefault}}{E_{totaldefault}} \right| \times 100\% \tag{7}$$

Figure 11 shows the total energy reduction rate D and average of the proposed technique for the total energy consumption of the default frequency governor for 15 workloads. Figure 11 shows the FPS reduction rate of the proposed low-power technique for the FPS of the default frequency governor while running each workload. The FPS of each technique is the average of N FPS collected at 50 ms data collection intervals as shown in $FPS_{average}$ of Equation (8).

$$FPS_{average} = \frac{\sum_1^N FPS_k}{N} \tag{8}$$

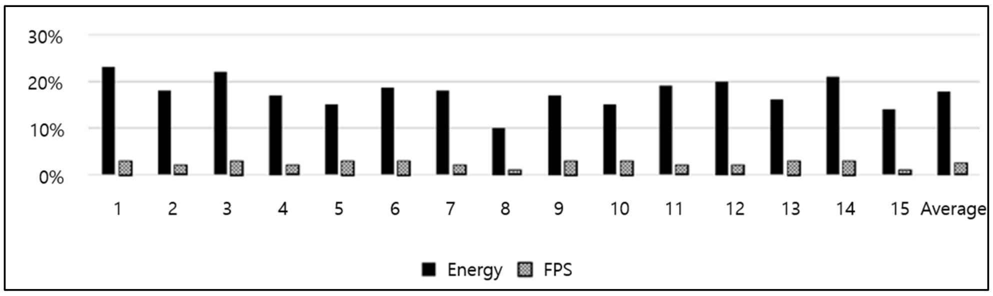


Figure 11. Energy reduction rate and FPS increase rate of the proposed method for the default frequency governor while running 15 workloads.

With the proposed low-power technique, the overall frequency is lower than that of the default frequency governor. Therefore, using the proposed low-power technique reduces the FPS along with the total energy consumption.

Using the low-power techniques proposed in Figure 11, compared to using the default frequency governor of the Samsung Galaxy S7 smartphone, the total energy consumption shows an average reduction rate of 18.11%, and the FPS shows an average reduction rate of 3.12%. The total energy consumption is greatly reduced while the FPS is only slightly reduced, showing an excellent power consumption reduction rate while maintaining QoS.

We measured CPU/GPU frequency by phase, measuring power consumption and FPS for workloads, and then changing CPU/GPU frequency for four workload bounces to measure power consumption and FPS rate tables according to the reference CPU/GPU frequency. In order to compare the cost-based and phase-based low-power techniques, the default frequency governor was selected.

Table 7 shows the energy reduction rate and FPS increase rate of the cost-based low-power technique and the phase-based low-power technique for the default frequency governor. The "cost" used in other low-power method studies [7] is a value that simply determines the CPU/GPU intensity of the workload. Additionally, the "phase" used in the method proposed in this paper is the execution state classified according to the usage ratio of various instructions in the processor occupied by the workload. Therefore, we compared the cost-based method and the phase-based method, seeing that phase can classify workloads more dynamically than cost. Additionally, through experiments, the phase-based method showed a greater energy reduction and a smaller FPS increase than the cost-based method, showing that using phase is more effective than cost.

Table 7. The energy reduction rate and FPS increase rate of the cost-based low-power technique and the phase-based low-power technique for the default frequency governor.

	Energy					**FPS**
	CPU Little	**CPU Big**	**GPU**	**Display**	**Total**	
Cost-based	1%	9%	10%	0%	10%	5%
Phase-based	1%	7%	23%	0%	18.11%	3.12%

According to Table 7, it can be seen that the cost-based low-power technique shows an energy reduction rate of 9% and the phase-based low-power technique shows an energy reduction rate of 7%, so the cost-based low-power technique is better. On the other hand, in a GPU, the cost-based low-power technique shows a 10% reduction rate and the phase-based low-power technique shows a 23% reduction rate, so the phase-based low-power technique is better. However, comparing actual energy consumption, GPU energy is generally larger than CPU energy, so in total, the phase-based low-power technique reduction rate is 18.11%, which is greater than the cost-based low-power technique's reduction rate of 10%. In the case of FPS, the phase-based low-power technique is 3.12%, which is less than 5% of the cost-based low-power technique. In the end, it could be verified that the phase-based low-power technique is superior to the cost-based low-power technique, both in energy consumption and in FPS.

6. Conclusions

In this paper, a novel phase-based CPU and GPU integrated low-power approach is proposed for 3D game applications running on Android devices. Compared to the default low-power techniques provided by the Linux kernel-based Android operating system and the existing low-power technique [7] using the CPU/GPU normalized cost based on utilization, the proposed method was found to achieve better power savings while mitigating performance loss though extensive experiments.

The proposed method in this paper showed only 3.12% of the FPS reduction rate compared to the interactive governor, one of the frequency governors provided in the Samsung Galaxy S7 smartphone, while the energy consumption reduction rate was 18.11%. In addition, its FPS reduction rate was 2% lower, and the energy consumption rate was 8% higher compared to when the normalized cost of HiCAP was used instead of phase prediction in the workload classification method for the proposed technique.

As future work, we plan to study high-performance low-power techniques for mobile devices for various high-performance applications, such as deep learning, virtual reality, and augmented reality. In addition, we intend to apply the proposed technique to various Android platform devices such as head mounted display (HMD) devices.

Author Contributions: Conceptualization, Y.-J.K.; software, S.-R.O.; validation, Y.K., Y.-J.K.; investigation, S.-R.O., Y.-J.K.; writing—original draft preparation, S.-R.O.; writing—review and editing, Y.K., Y.-J.K.; visualization, S.-R.O.; supervision, Y.-J.K.; project administration, Y.K.; funding acquisition, Y.-J.K. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Future Combat System Network Technology Research Center program of the Defense Acquisition Program Administration and Agency for Defense Development (UD190033ED).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. How Many People Have Smartphones in 2021? Available online: https://www.oberlo.com/statistics/how-many-people-have-smartphones (accessed on 15 June 2022).
2. Which Smartphone Features Really Matter to Consumers? Available online: https://blog.gwi.com/chart-of-the-week/smartphone-features-consumers/ (accessed on 15 June 2022).
3. Is Smartphone Battery Capacity Growing or Staying the Same? Available online: https://c.mi.com/thread-2085983-1-0.html?mobile=no (accessed on 15 June 2022).
4. Big Phone Batteries Don't Guarantee Long Battery Life. Available online: https://www.androidauthority.com/what-is-mah-smartphone-battery-life-1113391/ (accessed on 15 June 2022).
5. Lee, K.; Ohk, S.-R.; Lim, S.-G.; Kim, Y.-J. Phase-Based Accurate Power Modeling for Mobile Application Processors. *Electronics* **2021**, *10*, 1197. [CrossRef]
6. Brodowski, D. CPU Frequency and Voltage Scaling Code in the Linux (TM) Kernel. Available online: https://www.kernel.org/doc/html/latest/cpu-freq/index.html (accessed on 15 June 2022).
7. Park, J.G.; Dutt, N.; Kim, H.; Lim, S.S. HiCAP: Hierarchical FSM-based Dynamic Integrated CPU-GPU Frequency Capping Governor for Energy-Efficient Mobile Gaming. In *Proceedings of the 2016 International Symposium on Low Power Electronics and Design (ISLPED'16)*; Association for Computing Machinery: New York, NY, USA, 2016; pp. 218–223. [CrossRef]
8. Dietich, B.; Peters, N.; Park, S.; Chakraborty, S. Estimating the Limits of CPU Power Management for Mobile Games. In Proceedings of the 2017 IEEE International Conference on Computer Design (ICCD), Boston Area, MA, USA, 5–8 November 2017; pp. 1–8. [CrossRef]
9. Han, S.; Yun, Y.; Kim, Y.H.; Kang, S. Proactive Scenario Characteristic-Aware Online Power Management on Mobile Systems. *IEEE Access* **2020**, *8*, 69695–69711. [CrossRef]
10. Carvalho, S.A.L.D.; Cunha, D.C.D.; Silva-Filho, A.G.D. Autonomous Power Management for Embedded Systems Using a Non-linear Power Predictor. In Proceedings of the 2017 Euromicro Conference on Digital System Design (DSD), Vienna, Austria, 30 August–1 September 2017; pp. 22–29. [CrossRef]
11. Kwon, E.; Han, S.; Park, Y.; Yoon, J.; Kang, S. Reinforcement Learning-Based Power Management Policy for Mobile Device Systems. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2021**, *68*, 4156–4169. [CrossRef]
12. Hong, S.; Kim, S.; Kim, Y. LGC-DVS: Local Gamma Correction-Based Dynamic Voltage Scaling for Android Smartphones With AMOLED Displays. *IEEE J. Electron Devices Soc.* **2017**, *5*, 432–444. [CrossRef]
13. Park, J.-G.; Hsieh, C.-Y.; Dutt, N.; Lim, S.-S. Co-Cap: Energy-efficient cooperative CPU-GPU frequency capping for mobile games. In *Proceedings of the 31st Annual ACM Symposium on Applied Computing*; Association for Computing Machinery: New York, NY, USA, 2016; pp. 1717–1723. [CrossRef]
14. Hong, S.; Kim, S.-W.; Kim, Y.-J. 3 channel dependency-based power model for mobile AMOLED displays. In Proceedings of the 2017 54th ACM/EDAC/IEEE Design Automation Conference (DAC), Austin, TX, USA, 18–22 June 2017; pp. 1–6. [CrossRef]
15. Monsoon Solutions. Power Monitor. Available online: https://www.msoon.com/LabEquipment/PowerMonitor/ (accessed on 15 June 2022).
16. Ibrahim, M.; Hamarash, I. Dynamic voltage frequency scaling (DVFS) for microprocessors power and energy reduction. In Proceedings of the 4th International Conference on Electrical and Electronics Engineering, Bursa, Turkey, 7–11 December 2005.
17. Jung, H.; Gil, A.; Kim, J. A Case Study of Limited Dynamic Voltage Frequency Scaling in Low-Power Processors. World Academy of Science, Engineering and Technology, Open Science Index 108. *Int. J. Electr. Comput. Eng.* **2015**, *9*, 1523–1526.
18. Claypool, M.; Claypool, K.; Damaa, F. The Effects of Frame Rate and Resolution on Users Playing First Person Shooter Games. *Proc. SPIE Int. Soc. Opt. Eng.* **2006**, *6071*, 607101. [CrossRef]
19. 3Dmark—The Gamer's Benchmark. Available online: https://play.google.com/store/apps/details?id=com.futuremark.dmandroid.application (accessed on 16 June 2022).
20. Madagascar 3D Benchmark. Available online: https://play.google.com/store/apps/details?id=app.ext2dev4me.m3db (accessed on 16 June 2022).

21. Benchmark 3D Kassja Graphics. Available online: https://play.google.com/store/apps/details?id=com.mkdesignmobile.KassjaBenchmark (accessed on 16 June 2022).
22. Renderscript Benchmark. Available online: https://apkpure.com/kr/renderscript-benchmark/name.duzenko.farfaraway (accessed on 16 June 2022).
23. NenaMark2. Available online: https://apkpure.com/nenamark2/se.nena.nenamark2 (accessed on 16 June 2022).
24. V1—GPU Benchmark Pro. Available online: https://apkpure.com/kr/v1-gpu-benchmark-pro/com.InventedGames.V1Benchmark (accessed on 16 June 2022).
25. Seascape Benchmark—GPU Test. Available online: https://play.google.com/store/apps/details?id=com.nature.seascape (accessed on 16 June 2022).
26. Basemark GPU. Available online: https://play.google.com/store/apps/details?id=com.basemark.basemarkgpu.free (accessed on 16 June 2022).
27. 3D Benchmark. Available online: https://apkpure.com/kr/3d-benchmark-android-gamers/com.cabodidev.threedbenchmark (accessed on 16 June 2022).
28. Relative Benchmark. Available online: https://play.google.com/store/apps/details?id=com.re3.benchmark (accessed on 16 June 2022).
29. Nah, J.-H.; Suh, Y.; Lim, Y. L-Bench: An Android benchmark set for low-power mobile GPUs. *Comput. Graph.* **2016**, *61*, 40–49. [CrossRef]
30. ARM-Software/Gator. Available online: https://github.com/ARM-software/gator (accessed on 16 June 2022).
31. LineageOS/Herolte. Available online: https://wiki.lineageos.org/devices/herolte/ (accessed on 25 July 2022).