IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received June 3, 2019, accepted July 19, 2019, date of publication July 26, 2019, date of current version August 12, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2931290

A Practically Applicable Performance Prediction Model Based on Capabilities of Texture Mapping Units for Mobile GPUs

JUWON YUN[1], JINYOUNG LEE[1], CHEONG GHIL KIM[2], (Member, IEEE), YEONGKYU LIM[3], JAE-HO NAH[3], YOUNGSIK KIM[4], AND WOO-CHAN PARK[1]

[1]Department of Computer Engineering, Sejong University, Seoul 05006, South Korea
[2]Department of Computer Science, Namseoul University, Cheonan 31020, South Korea
[3]LG Electronics, Seoul 06772, South Korea
[4]Department of Game and Multimedia Engineering, Korea Polytechnic University, Siheung 15073, South Korea

Corresponding author: Woo-Chan Park (pwchan@sejong.ac.kr)

This work was supported in part by the Institute for Information and communications Technology Promotion through the Korean Government (MSIP) (Development of mobile GPU hardware for photo-realistic realtime virtual reality) under Grant 2016-0-00204, and in part by the National Research Foundation of South Korea (NRF) under the Korean Government (MSIP) under Grant 2019R1A2C1005163.

ABSTRACT The power consumption models of mobile application processors have emerged as key objects of interest following the tremendous growth in mobile device production given that such consumption is an important factor in the graphics performance of mobile technologies. Conventionally, the performance of the graphics processing units (GPUs) depends critically on texture mapping units, which is why the number of such GPU components and texture fill rates value prominently whenever the GPU performance is evaluated. Our previous work has established a model to predict maximum performance based on unified shaders. By extending the work, this paper developed a practically applicable GPU performance prediction model on the basis of texture mapping performance. The effects of increased texture mapping units on unified shader performance and GPU efficiency were examined, and a performance prediction model based on the number of frames per second (FPS) was constructed. For these purposes, a benchmark related to texture mapping units was formulated and the experiments were conducted to determine utilization factors that are relevant to GPU performance and efficiency. The final stage in model construction involved establishing a relationship between the previously investigated utilization factors and relevant resources that are consumed during graphics processing. The experimental results showed that the proposed prediction model produced an average error rate of 5.77%.

INDEX TERMS Computer graphics, dynamic voltage scaling, performance analysis, performance evaluation, prediction method.

I. INTRODUCTION

The performance of application processors (APs) in mobile devices and other embedded systems has dramatically improved owing to the intensive advancement of semiconductor processes and design technologies. Currently APs are being developed and manufactured with a single chip in which multicore central processing units (CPUs) and graphics processing units (GPUs) are integrated using system-on-chip technology. As a result, it is common that the performance of their CPU performance has come up to 2.84 GHz clock speed and 64-bit 8 cores. Along with the performance increase, more demand for graphics applications that require very advanced technologies of augmented and virtual reality and high-performance three-dimensional (3D) rendering are getting popular. Such high-quality graphics are processed at considerable computational cost and power consumption [1], thus prompting enhancements to graphics processing technology.

However, applying advanced graphics processing techniques on a resource-constrained mobile platform has become the most important issue to address. To improve high-quality graphics processing in a limited environment, mobile GPU manufacturers focus on developing chip designs that

The associate editor coordinating the review of this manuscript and approving it for publication was Xiaogang Jin.

enable efficient processing with low power consumption, and developers emphasize application optimization for smooth operation with minimal resources.

Typically, mobile APs have low power consumption with the help of low power design methodologies. However, when using a GPU, it consumes a lot of power in nature. In this case, that of the mobile AP is closely related to the use of them. Higher graphics quality and faster processing performance may inherently cause power consumption and heat generation. In this regard, current mobile AAA games support an option for user control over graphics quality, texture resolution, and frame rate. As the prediction of performance and power consumption of mobile GPU becomes important, various prediction methods are being developed. However, accurate power consumption prediction should be made by GPU performance.

In general, the models that can predict the power consumption of mobile GPUs have been studied on the basis of three methods. The first one measures and analyzes power consumption with reference to each phase of a graphics pipeline; the second one predicts power consumption by analyzing dynamic voltage and frequency scaling (DVFS) according to information on the utilizations of both CPU and GPU; the third one predicts power consumption using an equation extracted through learning. More details of these three methods are provided below.

The first method involves the use of primitive information on a 3D graphics pipeline [5], the collection of batch, vertex, and fragment data by each rendering path [6], and the analysis of power consumption by 3D graphics components in each pipeline stage [7], [8]. The second method entails the use of information on vertex-processing and pixel-processing loads [9], frequency scaling that accounts for user and application conditions [10], and a dynamic power predication scheme that accords with CPU and GPU usage [11]. The third method calls for a combination of data collection and online learning using GPU frequency and workload factor [12], machine learning that uses the profiling data of CPUs [13], and a linear model that uses game application as learning data [14].

The research of predicting the performance of mobile GPU based on power usage can be approached in four ways: 1) GPU performance prediction based on changes in GPU-related parameters [15]; 2) the measurement and computation of hardware performance in each graphics processing phase [16]; 3) calculate using equations that factor the performance of the GPU hardware [17]; 4) maximum performance prediction grounded in the processing performance of a unified shader [18].

The first method predicts the performance of a program that uses designated instances by changing parameters such as the number of threads, blocks, and streaming multiprocessors. The second one predicts performance by extracting a linear regression model when it learns about a GPU's low-level performance and frame rate. The third one compares the performance of a GPU with that of video graphics arrays in a proportion calculation scheme by converting various GPU computational performance levels into an equation. The final method determines the processing performance of a unified shader on the basis of the examined relationship among hardware units that constitute a GPU and predicts maximum performance by normalizing utilization values.

In our previous study [18], we identified the unified shader as a key component of GPU performance in which much of the graphics pipeline operations were performed. Based on this, we derived a new GPU performance prediction model by running it at maximum. More advanced than before, the performance prediction model proposed in this work has utilized a complex correlation between TMU performance and the maximum throughput of USI per GFLOPS in order to predict the FPS.

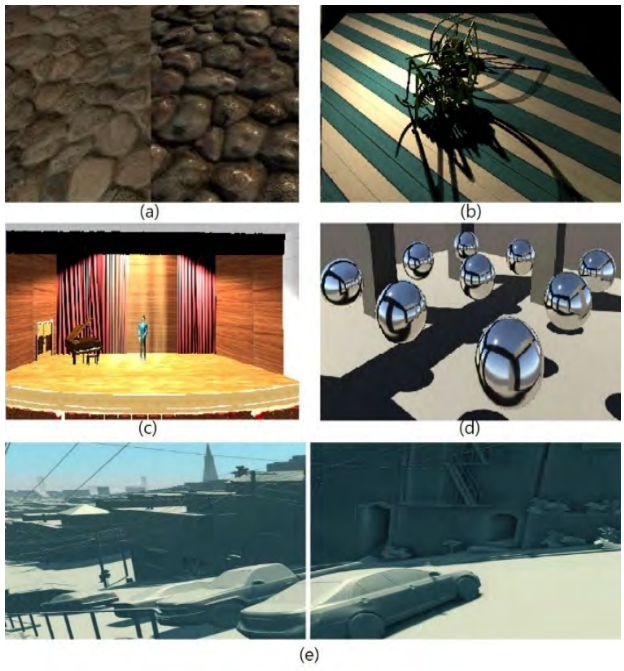

FIGURE 1. Texture mapping via advanced graphics processing techniques in a texture mapping unit; (a) Bump mapping off/on, (b) shadow mapping, (c) light mapping, (d) environment mapping, (e) screen space ambient occlusion off/on [19].

Figure 1 shows that for high-end graphics processing technology, TMUs perform texture mapping tasks, such as bump mapping, shadow mapping, light mapping, environment mapping, and screen space ambient occlusion. TMU performance is particularly important in high-performance GPUs wherein texture processing is several times more complicated than pixel processing. Considering that more high-end, high-quality texture mapping technologies will be used in the future, predicting TMU performance is a critical requirement.

The research confirmed that the processing efficiency of unified shader instruction (USI) increased as the unified shader, which is a core factor in mobile GPU performance, was affected by TMU texture processing. With reference to this result, the gigaflops (GFLOPS) consumed per frame was

TABLE 1. Comparison table of characteristics of performance related reference paper.

Category	[15]	[16]	[17]	[13]	[18]
Target	Desktop GPU	Mobile GPU	Desktop GPU	Desktop GPU	Mobile GPU
Analytic Model	Mathematical equation	Linear Regression	Mathematical equation	K-Nearest Neighbor, Support Vector Machine	Mathematical equation
Model Factor	The relevant parameters: The total number of resident blocks, number of threads in each block, the total number of blocks spawned in the device, the total number of streaming multi-processors available in the device	Low-level performance: Triangle Throughput, texture fill rate, the calculation speed of shaders (FLOPS)	Hardware specification: Number of shaders, texture mapping units, render output units, Memory bandwidth and GPU frequency Operation speed: Shader, texture, rendering	Magnitude of GPU speedups: Computation, memory, control flow, OpenMP, aggregate	Low-level performance: Instruction throughput on unified shader (vertex & fragment), GPU utilization, shaders busy, shader ALU capacity Hardware specification: GPU frequency, the calculation speed of shaders (GFLOPS)
Result	The prediction error: 0.13~5.69%	The maximum error rate: T-Rex 0.97 FPS, Egypt 0.93 FPS	None	The accuracy of at least: 77-90%, and the best device: 91%	The average error rate: 2.96% T-Rex (0.10~6.22%) MANHATTAN (0.95~4.32%) Bootcamp (0.02~5.91)

obtained, and a performance prediction model that predicts the number of FPS was developed. For the performance evaluation, the Snapdragon Profiler [20] was used and experiments based on the measured low-level performance were conducted. As a result, the GFLOPS, GPU efficiency, and GFLOPS per frame of the device are applied into the performance prediction model. The experimental results indicated that the average error rate between the predicted and measured values of the corrected FPS was 5.77%.

The rest of the paper is organized as follows. Chapter 2 describes the unified shader-based performance prediction model that was investigated in our previous study. Chapter 3 explains the GPU efficiency that is based on the number of TMUs in a device. Chapter 4 discusses the FPS-based performance prediction model that was constructed with reference to the relationship among TMUs, the throughput of a unified shader, and GPU efficiency. Chapter 5 presents the results of experiments on the model put forward in the current work, and Chapter 6 concludes the paper.

II. BACKGROUND

A. PERFORMANCE PREDICTION MODELS FOR GPUs

Table 1 classify the performance prediction models proposed in previous work, including our study, according to categories of characteristics. In the table, the ''target'' identifies the platform of the device to be predicted, ''analytic model'' refers to the model used for prediction, ''model factor'' refers to the factors used to derive the analytic models, and ''result'' represents the predicted value derived from the performance model.

Hasan *et al.* [15] developed an analytic model to predict GPU performance for computationally intensive tasks. The analytic model is based on varying the relevant parameters. These parameters consist of total number of threads, blocks, streaming multi-processors. These parameters are changed to predict the performance of the program for a given instance. Although this analytical model was developed for computationally intensive tasks on desktop GPUs, it may not be appropriate for mobile GPU environments that perform graphics-intensive tasks rather than computationally intensive tasks.

Xie *et al.* [16] proposed an estimation method for mobile GPUs. They used 3D rendering low-level performance which was measured at the early stage of SoC design. In addition, they built a linear regression model for estimating mobile GPUs. However, since the utilization factor which is variable due to DVFS is not considered, it may be difficult to predict performance close to actual performance.

Lee *et al.* [17] established a mathematical equation for predicting desktop GPUs. To estimate desktop GPUs, the author derived operation speeds that occurs during calculation in the graphics pipeline. In addition, the hardware specification of the desktop GPU (GPU frequency, shader performance, texture fill rate, memory bandwidth, pixel processing performance) was applied as an input parameter of the equation. The equation is computed in such a way as

to sum up the performance of the units being processed in the graphics pipeline. However, the bottleneck depends on the type of job being processed, the method of calculating the equation may not be suitable for performance prediction.

Baldini *et al.* [13] proposed a performance prediction model that could estimate speedup in a desktop GPU. This model derived algorithms for performance prediction through machine learning methods using CPU profiling data as input parameters. Unlike the mobile GPU, the power consumption and heat generation of the desktop GPU are controlled by the user's OS setting. Because of this, DVFS and throttling functions were not considered as factor of limiting performance. Therefore, the performance prediction model may be inapplicable to mobile GPUs.

B. OUR PREVIOUS WORK: PERFORMANCE PREDICTION MODEL WITH USI

In our previous study, we presented a new approach to the prediction of mobile GPU performance by analyzing the relationship between hardware units that make up a GPU and GPU performance and ascertaining the maximum performance for which DVFS and throttling are considered [18]. Through micro-benchmarking, we were able to measure low-level performance values and investigate their direct impact on performance by assigning workloads to each hardware device. The simulation results showed that a unified shader, which accounted for the largest space among the hardware units that composed a GPU, directly influenced performance. On this basis, we constructed a performance prediction model, with the instruction throughput in unified shader (USI) and predicted the USI per second throughput.

The simulation results in [18] showed the unstable performance outcomes each time because of the throttling that forcibly diminishes performance by the DVFS and temperature of a device. We have investigated the relationship between GPU performance and utilization through issues related to DVFS and throttling.

In the graphics pipeline, the utilization values of each unit can change in real time depending on the task being given. Because of this issue causes the processing speed of each unit to vary and it is difficult to predict GPU performance accurately. To address the issue, Yun *et al.* [18] elevated the utilization value of a GPU to its maximum level when they developed a GFLOPS-based USI_{Max} model; the USI_{Max} refers to the maximum performance in which instructions in a unified shader can be processed.

Accurately predicting performance is very difficult because utilization values change in real time in accordance with a use environment, thus producing different processing speeds whenever a GPU is used. To address the issue, Yun *et al.* [18] elevated the utilization value of a GPU to its maximum level when they developed a GFLOPS-based USIMax model.

Thus, enabling us to explore the relationship between the GFLOPS of the GPU shader core and GPU performance and extracted the corresponding equation (Figure 2).

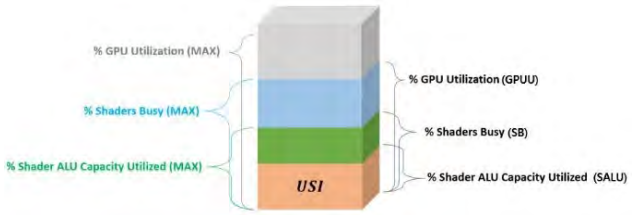

FIGURE 2. The factors on the right of the figure reflect actual performance, and those on the left indicate maximum performance. This figure is the same as Figure 9 in [13].

TABLE 2. The parameters used in this paper.

Category	Feature	Mnemonic
Instruction throughput	USI throughput/second	USI
	USI throughput to the maximum	USI_{Max}
	USI throughput/second (Corrected GPU utilization value to 100%)	$USI_{Corrected}$
Utilization	GPU utilization	GPUU
	Shaders busy	SB
	Shader ALU capacity utilized	SALU
Hardware specification	Shader performance	GFLOPS
	GFLOPS of the device	GF_{Device}
	GFLOPS per frame	GF_{Frame}

Equations have been validated in benchmark experiments on mobile devices. This model predicted maximum performance with an average error of 3.32%.

III. IMPROVED PERFORMANCE AND EFFICIENCY FOLLOWING INCREASED NUMBER OF TMUs

A. TMU AND GPU EFFICIENCY

Next to unified shaders, TMUs process the largest amount of work in a graphics pipeline. In the fragment shader phase that follows vertex shader work, texture mapping and memory access tasks for texture map loads are processed in TMUs. If processing occurs fast because of the large number of units that carry out texture mapping, then the delay in the work processing of a unified shader decreases. The efficiency of the shader core increases as the number of instructions that are processed rises after delay reduction in unified shader processing. Because the augmented efficiency of the shader core increases throughput, the overall processing performance of a graphics pipeline improves.

To confirm the improvement in shader efficiency following an increase in the number of TMUs in a device, an experiment involving GFXBench (an application generally used to measure the performance of mobile devices) with textures was conducted. Among the experimental devices used, the GPU in G4 and G3 are the Adreno 4 series and are based on similar GPU architectures. The performance comparison in Table 3 shows that the two models have identical characteristics, except for the number of TMUs. To verify the TMU-induced improvement in the performance of those models, the low-level performance values of related items in GFXBench were measured using the Snapdragon Profiler.

TABLE 3. Comparison of G4 and G3 CAT 6 performance.

	GPU clock (MHz)	Render output unit (ROP)	TMU	Shader performance 16 bit (GFLOPS)	Shader performance 32 bit (GFLOPS)	Product
Adreno 418	600	8	8	316.8	163.2	G4
Adreno 420	600	8	16	326.4	172.8	G3 CAT 6

TABLE 4. MANHATTAN and TREX benchmark experiment results of G4 and G3 CAT6.

Scene	GPUU (%)		SB (%)		SALU (%)		Actual USI throughput		FPS	
	G4	G3 CAT6	G4	G3 CAT6	G4	G3 CAT6	G4	G3 CAT6	G4	G3 CAT6
MANHATTAN	95.68	69.10	71.20	72.51	27.92	39.92	16.2G	17.0G	16	16
TREX	99.75	90.11	72.51	69.14	25.13	32.81	14.2G	16.8G	36	37

The measured FPS performance values of them were very similar, but the GPU utilization and shader ALU capacity consumed were remarkably different shown in Table 4. The experimental results on the MANHATTAN scene showed that G4 exhibited a performance of 16 FPS while running at a GPUU of 95.68%. G3 CAT6 did a performance of 16 FPS while operating at a GPUU of 69.1%. For the TREX scene, G3 CAT6 ran at a smaller level of GPU utilization but achieved a performance similar to that of G4. As a result, it is confirmed that the processing efficiency of the GPU increases when processing with similar frame rate performance is performed with a relatively small GPU frequency.

B. IMPROVEMENT IN USI PROCESSING PERFORMANCE THANKS TO TMUs

Due to fluctuating GPU utilization values, it is difficult to pinpoint performance differences due to TMU. To solve this problem, the test environment was designed to approximate conditions wherein GPU frequency is at its maximum level of 600 MHz; this approximation was achieved by correcting the GPU utilization value to its maximum. Normalization to identical conditions via such correction enabled the monitoring of the TMU-driven increase in the USI performance of the experimental devices. The normalization was performed using the following equation:

$$USI_{Corrected}(GPUU\,100\%) = USI \times 100/GPUU \qquad (1)$$

In (1), USI is the weighted sum of the number of instructions that are processed in the vertex shader of a unified shader and the number of instructions that are processed in the fragment shader. This weighted sum is calculated taking into account the percentage of the work done by the vertex shader and the fragment shader. GPUU refers to the measured GPU utilization value. So (1) is used to compute the corrected USI throughput when GPU frequency is at its maximum.

TABLE 5. Corrected USI throughput (GPUU 100%) of GFXBench.

USI throughput when GPU utilization is 100%			
	G4	G3 CAT6	Increase of performance (Number of TMUs increased)
MANHATTAN	16.9 G	24.6 G	1.4556 (45.56%)
TREX	14.2 G	18.7 G	1.3098 (30.98%)

As shown in Table 5, the corrected USI throughput of G3 CAT6, which has twice the number of TMUs, is approximately 45.5% and 30.9% higher than that of G4 for the MANHATTAN and TREX scenes, respectively. The average performance increase of 38.2% can be attributed to the increased USI throughput due to TMUs rather than the performance indicator GFLOPS of shader performance. This increase reflects a difference of 9.6 GFLOPS between G3 CAT6 and G4. To confirm these findings, we have fabricated and experimented a microbenchmark without textures.

The microbenchmark in Table 6 was developed in such a way that the corrected USI throughput could be displayed by increasing the number of vertices from 0.1 million to 0.9 million (Figure 3). The microbenchmark results showed that the corrected USI throughput of G3 CAT6 increased by 9.15% over the level derived by G4. The findings also indicated a large difference from the results of the GFXBench experiment with respect to the increase in average corrected USI throughput. Because there is no texture in the benchmark in Table 6, the USI correction throughput is increased by about 9.15% due to the shader performance GFLOPS difference of 9.6, regardless of the number of TMUs.

The non-textured experiment also confirmed that the efficiency of the shader core increased when the number of TMUs increased. This phenomenon is attributed to the fact

TABLE 6. Corrected USI throughput (GPUU 100%) of the microbenchmark without textures.

Corrected USI throughput (GPUU 100%){VI × weight) + FI} ÷ GPU utilization			
Vertices	G4	G3 CAT6	Increase of performance: G4 → G3 CAT6
100k	7.02 G	8.01 G	1.140549236
200k	7.24 G	9.16 G	1.264193623
300k	8.16 G	9.22 G	1.129668782
400k	8.14 G	9.52 G	1.169785576
500k	9.07 G	9.75 G	1.075155825
600k	9.57 G	9.90 G	1.034325676
700k	10.05 G	10.19 G	1.014561497
800k	10.14 G	10.25 G	1.010068068
900k	10.54 G	10.39 G	0.985938119
Average	8.88 G	9.60 G	**1.091582934 (9.15%)**

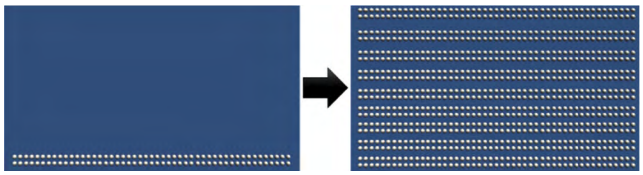

FIGURE 3. Microbenchmark where the vertices increased from zero to a range of 0.1 to 0.9 million.

that under these conditions, delayed work processing in a unified shader decreases and instruction processing is executed more efficiently. On this basis, we conclude that USI processing performance in general applications with textures can be affected by TMU performance.

IV. PROPOSED FPS-BASED PERFORMANCE PREDICTION MODEL

The difference in $USI_{Corrected}$ demonstrated that the instruction throughput in a unified shader ultimately increases as the efficiency of the shader core improves. In order to propose an FPS-based performance prediction model, the USI throughput increase rate according to the number of TMUs is defined as ETMU and the GPU efficiency value is defined as EGPU. Predicting FPS performance necessitates determining the EGPU, ETMU, and shader GLFOPS of an experimental device and the GFLOPS required to draw one frame. The USI value that is processed to draw one frame was obtained first to identify the GFLOPS required to draw such frame. The following equation was employed in calculating the USI per frame:

$$USI_{Frame} = USI_{Corrected} \div FPS_{Corrected} \quad (2)$$

$$GF_{Frame} = USI_{Frame} \div USI_{GFLOPS} \quad (3)$$

The $FPS_{Corrected}$ derived using equation (2) refers to the FPS value obtained when GPU frequency is corrected to its maximum level in an experimental benchmark scene. In (3), the USI_{GFLOPS} is the USI throughput per 1 GFLOPS. We derived USI_{GFLOPS} value of approximately 0.5137G through experiment in our previous work [18]. The equation yields the USI value that is processed to draw one frame. The GF_{Frame} that is consumed to draw one frame can be obtained by dividing the computed USI per frame by the USI average value that is processed in each GFLOPS, as shown in (Table 7). In the experiment (V), the average GFLOPS values per frame of the devices are 1.94 and 0.82 for the MANHATTAN and TREX scenes, respectively. Also, in the BOOTCAMP scene of Unity sample project, the GFLOPS values per frame value is 1.14. The MANHATTAN scene requires much higher GFLOPS to draw one frame than the TREX scene. This indicates that the MANHATTAN scene has higher scene complexity and shader complexity than the TREX scene. The GPU efficiency of the experimental devices should be applied to predict FPS values on the basis of the GFLOPS values obtained above. Among the GPU efficiency values, the EGPU is computed using (4).

$$EGPU = USI_{Corrected} \div USI_{Max} \quad (4)$$

In (4), the USI_{Max} denotes the maximum level of USI that can be processed in mobile GPUs. EGPU is obtained by dividing the corrected throughput by the maximum USI throughput. ETMU pertains to the additionally improved GPU efficiency stemming from the increased number of TMUs in mobile GPU series that have similar architectures. For example, In the MANHATTAN scene (Table 7), GPU efficiency was 18.09% on G4, which has a built-in Adreno 400 series. GPU efficiency increased by 28.95% on G3 CAT6, which is equipped with the same Adreno 400 series. Of these values, 10.86% is the ETMU value, which is additionally increased by an increase in the number of TMUs. The GPU efficiency value, which is one of the factor values for predicting the FPS, can be obtained through (4). Finally, the prediction target for FPS is determined thus:

$$FPS = \{GF_{Device} \times (EGPU + ETMU)\} \div (GF_{Frame}) \quad (5)$$

The FPS prediction value can be obtained from (5). The computed FPS is the value derived when GPU frequency is at its maximum level. GPU utilization data are normalized because performance differs by device given variances in DVFS.

V. SIMULATION

A. EXPERIMENTAL ENVIRONMENT

This section details the simulations conducted on the FPS-based performance prediction model. The used mobile devices, an LG G4, G3 CAT6, GFLEX2, and G5 have the maximum GFLOPS values of 162.3, 172.8, 326.4, and 339.4, respectively [21]. We took advantage of GFXBench and Unity BOOTCAMP scenes for simulation and verification. GFXBench is a commonly used benchmarking application consisting of MANHATTAN and TREX scenes. The BOOTCAMP scene is one of Unity sample projects and designed as an actual 3D game application.

TABLE 7. Results of GFXBench scenes and unity bootcamp scene for experimental devices.

GFXBench MANHATTAN	Corrected USI throughput (GPUU 100%)	Maximum USI throughput (GPUU 100%, SB 100%, SALU 100%)	GPUU(%)	SB(%)	SALU(%)	GPU efficiency	
						EGPU(%)	ETMU(%)
G4	16.9 G	83.2 G	95.68	71.20	27.92	18.17	-
G3 CAT6	24.6 G	85.0 G	69.10	72.51	39.92	18.09	10.86
G FLEX2	30.2 G	168.1 G	74.89	67.90	26.52	18.01	-
G5	47.1 G	175.8 G	99.69	75.52	35.53	26.83	-

GFXBench TREX	Corrected USI throughput (GPUU 100%)	Maximum USI throughput (GPUU 100%, SB 100%, SALU 100%)	GPUU(%)	SB(%)	SALU(%)	GPU efficiency	
						EGPU(%)	ETMU(%)
G4	14.2 G	81.7 G	99.75	72.51	25.13	19.02	-
G3 CAT6	18.7 G	82.5 G	90.11	69.14	32.81	17.07	5.62
G FLEX2	24.2 G	160.6 G	82.44	62.53	24.17	15.11	-
G5	33.5 G	177.1 G	99.39	74.09	25.55	18.93	-

Unity BOOTCAMP	Corrected USI throughput (GPUU 100%)	Maximum USI throughput (GPUU 100%, SB 100%, SALU 100%)	GPUU(%)	SB(%)	SALU(%)	GPU efficiency	
						EGPU(%)	ETMU(%)
G4	20.89G	82.2G	33.38	66.09	38.42	25.39	-
G3 CAT6	22.29G	82.9G	57.18	63.80	42.14	21.30	5.59
G FLEX2	28.70G	166.7G	53.68	52.14	33.00	17.21	-
G5	45.51G	170.5G	39.68	72.68	36.72	26.69	-

FIGURE 4. Benchmark scenes for verification. (a) GFXBench - manhattan scene (b) GFXBench - TREX scene (c) 3D game application Unity BOOTCAMP scene.

However, it is difficult to measure accurately performance for the reason that the resolution and density of experimental devices are different. As a result, we set the experiment environment to make the resolution and density of those devices equal for accurate performance measurement. The measurements are made by using the Snapdragon profiler, measuring 10 times for each benchmark, and then calculating the average value of them.

B. EXPERIMENTAL RESULTS

Table 8 compares the predicted values of the FPS-based model with the measured FPS values that corrected the GPU utilization by 100%. The difference between the predicted and measured FPS values in MANHATTAN and BOOTCAMP scenes are approximately 1.3 and 1.65, respectively, indicating that the predicted value and the measured value are almost coincides.

Table 8 shows that there is a variance of predicted error rate values when rendering different scenes on the same device. This is because each scene has different GFLOPS values for drawing one frame depending on various factors such as shader complexity, the OpenGL ES version used for rendering, texture types, and filtering methods. For this reason, the error rate differs depending on the characteristics

TABLE 8. Benchmark test results of the FPS-based performance prediction model.

GFXBench MANHATTAN scene	FPS	FPS correction (GPUU 100%)	FPS model outcomes (GPUU 100%)	Error FPS	Error Rate
G4	16	16.72	15.23	1.49	8.92%
G3 CAT6	16	23.15	25.70	2.55	10.99%
G FLEX2	22	29.37	30.20	0.82	2.8%
G5	47	47.14	46.78	0.37	0.78%

GFXBench TREX scene	FPS	FPS correction (GPUU 100%)	FPS model outcomes (GPUU 100%)	Error FPS	Error Rate
G4	36	36.09	37.75	1.66	4.60%
G3 CAT6	40	44.39	47.68	3.29	7.41%
G FLEX2	47	57.01	59.98	2.97	5.21%
G5	91	91.56	78.13	13.43	14.67%
Unity BOOTCAMP scene	**FPS**	**FPS correction (GPUU 100%)**	**FPS model outcomes (GPUU 100%)**	**Error FPS**	**Error Rate**
G4	13	38.95	39.90	0.95	2.45%
G3 CAT6	24	41.97	44.73	2.76	6.57%
G FLEX2	28	52.16	54.09	1.93	3.7%
G5	35	88.20	87.21	0.98	1.13%

of each scene if the scene is rendered differently even though on the same experimental device.

In the case of TREX, G5 has a performance difference of about 13.43 FPS compared to the prediction result. This is due to the visibility performance improved and the ALU/texture array efficiency due to the changes in the G5 architecture using the Adreno 500 series, unlike other devices as the Adreno 400 series [22]. The MANHATTAN scene and BOOTCAMP scene were created using OpenGL ES 3.0, whereas the TREX scene was produced using OpenGL ES 2.0. Similar to MANHATTAN, many current mobile AAA games are developed in versions higher than OpenGL ES 3.0.

The average error FPS value of the FPS-based model in comparison with the measured value of the corrected FPS was 2.76, shown in Table 8. When the error FPS value was converted into a percentage, the average error rate of the experimental devices was 5.77% which demonstrated the applicability of the model put forward in this work.

C. EXPERIMENTAL ANALYSIS

The prediction error results of the previous works [13], [15]–[18] are shown in Table 1. Comparisons between the results of this paper and previous works are as follows. As mentioned in the background, the performance prediction models of [13], [15]–[18] do not consider the TMU. Furthermore, [15], [17] and [13] are not included for comparison because they are targeting desktop GPUs. Hansan *et al.* [15] is an analytical model for computationally intensive tasks in a desktop GPU and is not suitable for mobile environments that perform graphic-centric tasks. Lee *et al.* [17] utilizes only the specifications of the desktop GPU and does not reflect the architecture and application characteristics. Baldini *et al.* [13] does not apply to mobile GPU because it does not reflect DVFS and throttling functions.

Xie *et al.* [16] predicts the performance of mobile GPUs using a correlation between high-level and low-level benchmarks. GFXBench TREX and EGYPT scenes were used for verification. As a result of the model prediction, the maximum FPS error value was 0.97 and 0.93 in each scene. When the error value is converted into a percentage, the average error rate of the experimental devices is 19.52% and 5.78%, respectively. The model may not be appropriate for comparing them on completely different GPU architectures. In fact, Anandtech's benchmark results [22] show that the low-level and high-level benchmarks results from GFXBench are not entirely proportional. Furthermore, the model was established at the peak of GPU performance without reflecting GPU utilization, while our proposed model reflected utilization values.

Our previous study [18] presented a more precise model using the numerical values obtained directly from the GPU profiler and showed a lower error rate than the existing methods; but it has a limitation that only USI_MAX value

is predicted. On the other hand, the proposed model in this paper predicts actual FPS, which is more progressive than our previous models.

The GFXBench TREX scene was commonly used in [16], [18] as well as our environment for simulation. Thus, we are able to directly compare the average error rate of each model in the scene: 19.52% in [16], 3.08% in [18], and 7.97%. Our previous work [18] reveals the lowest average error rate. But the predicted value is the maximum performance of the unified shader, which is not a model considering the actual FPS value. In contrast, our model is based on actual FPS prediction, so it could be understood that 7.97% may be numerically reliable.

VI. CONCLUSION

This research proposed a model that predicts the performance of mobile GPUs by deriving FPS, which could represent practical performance levels on the basis of GPU architecture and utilization information. The performance of a unified shader and the efficiency of a GPU increased following a rise in the number of TMUs in the experiment on a texture-related microbenchmark. On the basis of the results, an FPS prediction equation was derived by establishing the relationship between utilization factors that were relevant to GPU performance and efficiency and related resources consumed during graphics processing. This study predicted the FPS of GFXBench using the proposed model, which derived an average prediction error rate of 5.77% for all the experimental devices. The average error rate confirmed the applicability of the performance prediction model presented in this work. Another advantage of the proposed model is its ability to predict performance more practical than can an existing performance prediction model based on maximum USI throughput [18].

The experiments to proposed performance prediction model was performed only on Qualcomm Adreno GPUs. However, we believe that our model is applicable to other commodity mobile GPUs that include fundamentally similar rendering pipelines for OpenGL ES compatibility.

Future work will verify this model on various GPU architectures. Although our previous study [18] and the current work focus on shader units and TMUs (two elements with the largest influence on the performance of mobile GPUs), other subsystems in a GPU (geometry units, raster operations pipelines, memory units, GPU drivers, etc.) can also affect performance [23]. If we extend the current study to a performance model that can encompass various GPU subsystems, more accurate hardware modeling will be possible.

Furthermore, in mobile devices, power consumption is important as well as performance, and the power of a mobile GPU varies flexibly depending on the rendering tasks and performance required by the application. To exactly measure power consumption of a GPU, the separated power rail of the GPU on a development board is required as experimented in [24]; but unfortunately, it is hard to directly apply this method to commodity mobile devices.

Instead, GPU power consumption can be approximately compared by measuring system power consumption as Anandtech did; of course, this approach may show somewhat inaccurate results due to power consumption of other components such as a CPU, RAM, etc.

Regarding these points, we would like to investigate how to accurately estimate the GPU portion in entire system power consumption. Furthermore, we would like to extend our current model to a power-performance estimation model as future work.

REFERENCES

[1] R. Wang, B. Yu, J. Marco, T. Hu, D. Gutierrez, and H. Bao, "Real-time rendering on a power budget," *ACM Trans. Graph.*, vol. 35, no. 4, 2016, Art. no. 111. doi: 10.1145/2897824.2925889.

[2] X. Ma, Z. Deng, M. Dong, and L. Zhong, "Characterizing the performance and power consumption of 3D mobile games," *IEEE Computer*, vol. 46, no. 4, pp. 76–82, Apr. 2013. doi: 10.1109/MC.2012.190.

[3] T. Akenine-Möller and B. Johnsson, "Performance per What?" *J. Comput. Graph. Techn.*, vol. 1, no. 1, pp. 37–41, 2012.

[4] B. Johnsson, P. Ganestam, M. Doggett, and T. Akenine-Möller, "Power efficiency for software algorithms running on graphics processors," in *Proc. 4th ACM SIGGRAPH/Eurograph. Conf. High-Perform. Graph. (EGGH-HPG)*, vol. 12. Aire-la-Ville, Switzerland: Eurographics, 2012, pp. 67–75.

[5] J. M. Vatjus-Anttila, T. Koskela, and S. Hickey, "Power consumption model of a mobile GPU based on rendering complexity," in *Proc. 7th Int. Conf. Next Gener. Mobile Apps, Services Technol. (NGMAST)*, Sep. 2013, pp. 210–215.

[6] Y. Zhang, M. Ortin, V. Arellano, R. Wang, D. Gutierrez, and H. Bao, "On-the-fly power-aware rendering," *Comput. Graph. Forum*, vol. 37, no. 4, pp. 155–166, Jul. 2018.

[7] C.-W. Huang, Y.-A. Chung, P.-S. Huang, and S.-L. Tsao, "High-level energy consumption model of embedded graphic processors," in *Proc. IEEE DSP*, Jul. 2015, pp. 105–109.

[8] B. Mochocki, L. Lahiri, and S. Cadambi, "Power analysis of mobile 3D graphics," in *Proc. Conf. Design, Autom. Test Eur.*, Mar. 2006, pp. 502–507.

[9] T. Jin, S. He, and Y. Liu, "Towards accurate GPU power modeling for smartphones," in *Proc. 2nd Workshop Mobile Gaming*, Florence, Italy, May 2015, pp. 7–11. doi: 10.1145/2751496.2751502.

[10] W.-M. Chen, S.-W. Cheng, P.-C. Hsiu, and T.-W. Kuo, "A user-centric CPU-GPU governing framework for 3D games on mobile devices," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design*, Austin, TX, USA, Nov. 2015, pp. 224–231.

[11] Y. G. Kim, M. Kim, J. M. Kim, M. Sung, and S. W. Chung, "A novel GPU power model for accurate smartphone power breakdown," *ETRI J.*, vol. 37, no. 1, pp. 157–164, 2015.

[12] U. Gupta, J. Campbell, U. Y. Ogras, R. Ayoub, M. Kishinevsky, F. Paterna, and S. Gumussoy, "Adaptive performance prediction for integrated GPUs," in *Proc. 35th Int. Conf. Comput.-Aided Design*, Austin, TX, USA, Nov. 2016, pp. 1–8. doi: 10.1145/2966986.2966997.

[13] I. Baldini, S. J. Fink, and E. Altman, "Predicting GPU performance from CPU runs using machine learning," in *Proc. IEEE SBAC-PAD*, Oct. 2014, pp. 254–261.

[14] A. Pathania, A. E. Irimiea, A. Prakash, and T. Mitra, "Power-performance modelling of mobile gaming workloads on heterogeneous MPSoCs," in *Proc. ACM/EDAC/IEEE DAC*, Jun. 2015, pp. 1–6.

[15] K. S. Hasan, A. Chatterjee, S. Radhakrishnan, and J. K. Antonio, "Performance prediction model and analysis for compute-intensive tasks on GPUs," in *Proc. 11th IFIP Int. Conf. Netw. Parallel Comput.*, in Lecture Notes in Computer Science, Sep. 2014, pp. 612–617.

[16] Z. Xie, Y. Zhang, and L. Shi, "A method for estimating the 3D rendering performance of the SoC in the early design stage," *IEICE Electron. Express*, vol. 11, no. 11, 2014, Art. no. 20140386. doi: 10.1587/ELEX.11.20140386.
[17] D. Lee. *VGA Calculator*. Accessed: Mar. 3, 2016. [Online]. Available: http://udteam.tistory.com/59
[18] J. Yun, J. Lee, C. G. Kim, Y.-K. Lim, J.-H. Nah, Y. Kim, and W.-C. Park, "A novel performance prediction model for mobile GPUs," *IEEE Access.*, vol. 6, pp. 16235–16245, Mar. 2018.
[19] Bugraphicslab. *Screen Space Ambient Occlusion*. Accessed: May 20, 2018. [Online]. Available: https://bugraphicslab.wordpress.com/2011/07/19/screen-space-ambient-occlusion-ssao/
[20] Qualcomm, San Diego, CA, USA. *Snapdragon Profiler*. Accessed: Sep. 20, 2016. [Online]. Available: https://developer.qualcomm.com/software/snapdragon-profiler
[21] Wikipedia Contributors. *Adreno*. Accessed: Apr. 23, 2017. [Online]. Available: http://en.wikipedia.org/wiki/Adreno
[22] R. Smith and A. Frumusanu. *The Qualcomm Snapdragon 820 Performance Preview: Meet Kryo*. Accessed: Dec. 10, 2015. [Online]. Available: https://www.anandtech.com/show/9837/snapdragon-820-preview/4
[23] J.-H. Nah, Y. Suh, and Y. Lim, "L-Bench: An Android benchmark set for low-power mobile GPUs," *Comput. Graph.*, vol. 61, pp. 40–49, Dec. 2016.
[24] J.-H. Nah, B. Choi, and Y. Lim, "Classified texture resizing for mobile devices," in *Proc. ACM SIGGRAPH Talks*, 2018, Art. no. 73.

JUWON YUN was born in South Korea, in 1986. He received the B.S. degree from the Department of Game and Multimedia Engineering, Korea Polytechnic University, Siheung, South Korea, in 2013. He is currently pursuing the Ph.D. degree in computer engineering with Sejong University, Seoul, South Korea. His current research interests include sound tracing, game engine, mobile GPU, and computer graphics.

JINYOUNG LEE was born in South Korea, in 1988. He received the B.S. degree from the Department of Information Control Engineering, Kwangwoon University, Seoul, South Korea, in 2013. He is currently pursuing the Ph.D. degree in computer engineering with Sejong University, Seoul. His current research interests include computer graphics, computer architecture, and machine learning.

CHEONG GHIL KIM received the B.S. degree in computer science from the University of Redlands, CA, USA, in 1987, and the M.S. and Ph.D. degrees in computer science from Yonsei University, South Korea, in 2003 and 2006, respectively. He is currently a Professor with the Department of Computer Science, Namseoul University, South Korea. His research interests include multimedia embedded systems, mobile AR, and 3D contents.

YEONGKYU LIM received the B.S. degree from Kyungpook National University, in 1997, the M.S. degree from the Department of Computer Science, Korea University, in 1999, and the Ph.D. degree from the Department of Computer Science, Yonsei University, Seoul, South Korea, in 2013. He has been with LG Electronics, since 1999. His research interests include embedded systems, HCI, GPU architecture and benchmark, virtual reality, and GPGPU.

JAE-HO NAH received the B.S., M.S., and Ph.D. degrees from the Department of Computer Science, Yonsei University, in 2005, 2007, and 2012, respectively. He is currently a Graphics Professional with LG Electronics. He has coauthored more than 15 technical papers in the international journals and conferences, such as ACM TOG, IEEE TVCG, CGF, and HPG. His research interests include ray tracing, rendering algorithms, and graphics hardware.

YOUNGSIK KIM received the B.S., M.S., and Ph.D. degrees from the Department of Computer Science, Yonsei University, South Korea, in 1993, 1995, and 1999, respectively. He was with the System LSI, Samsung Electronics Company Ltd., as a Senior Engineer, from August 1999 to February 2005. Since March 2005, he has been with the Department of Game and Multimedia Engineering, Korea Polytechnic University. His research interests include 3D graphics and multimedia architectures, game programming, and SOC designs.

WOO-CHAN PARK received the B.S., M.S., and Ph.D. degrees in computer science from Yonsei University, Seoul, South Korea, in 1993, 1995, and 2000, respectively, where he was a Research Professor, from 2001 to 2003. He is currently a Professor of computer engineering with Sejong University, Seoul. His current research interests include ray tracing processor architecture, 3D rendering processor architecture, real-time rendering, advanced computer architecture, computer arithmetic, lossless image compression hardware, and application-specific integrated circuit design.