A 130.3mW 16-core Mobile GPU with Power-aware Approximation Techniques

Yu-Jung Chen [*1], Shan-Yi Chuang[2], Chung-Yao Hung[1], Chao-Hsien Hsu[1], Chia-Ming Chang[1],
Shao-Yi Chien [#1], Liang-Gee Chen[2]
[1]Media IC and System Lab and [2]DSP/IC Design Lab
Graduate Institute of Electronics Engineering and Department of Electrical Engineering
National Taiwan University, Taipei, Taiwan
Email:[*]d98943001@ntu.edu.tw, [#]sychien@cc.ee.ntu.edu.tw

***Abstract*—In this work, a 130mW 16-core mobile GPU is fabricated with TSMC 45nm technology. With three approximation techniques, this proposed architecture can trade-off between power consumption and visual quality to provide power-aware ability. Implementation results show that, with satisfactory visual quality, 53% of the power consumption can be reduced with the proposed *Approximated Precision Shader architecture* and *Screen-space Approximated Lighting technique*. Moreover, the *Approximated Texturing* technique reduces 24.57% L1 cache requests in our evaluation.**

I. Introduction

Different from desktop GPUs, the power consumption is one the most important design issues for mobile GPUs. There are many previous works of architecture- and circuit-level optimization of mobile GPUs [1, 2, 3]. Among them, Chang *et al.* propose an approximated rendering technique, where optimization across algorithm and architecture levels are considered [3]. This technique exploits the trade-off between rendered image quality and power consumption to achieve a better power efficiency. However, the image quality drops significantly when the low-power mode is selected.

Inspired by the work of Chang *et al.*, a low-power 16-core GPU is developed by comprehensively investigating the approximation techniques, that is, the technique that can better approximate the rendered results with lower power consumption. This chip is featured by the three proposed approximation techniques as follows. 1) *Approximated Precision Shader (APS)* architecture is a heterogeneous multi-core architecture, where low power consumption is achieved by appropriate workload distribution between complex and simple shader cores. 2) *Approximated Texturing (AT)* mechanism reduces the memory bandwidth of texture request by adaptively approximating the texture mipmapping with the concept of wavelet transform. 3) *Scree-space Approximated Lighting (SSAL)* technique can approximate the illumination results with adaptive sampling patterns and plane fitting. The proposed mobile GPU can achieve low power consumption while satisfactory visual quality is well maintained with these approximation techniques.

II. Architecture Overview

The architecture of the proposed 16-core mobile GPU is illustrated in Fig. 1. It is mainly composed of four shader clusters. Each shader cluster contains four unified shader cores, a texture unit and a corresponding texture L1 cache. Therefore, there are sixteen concurrent active threads operating in parallel, and each unified shader core adopts a four-channel SIMD processor architecture. The proposed *Approximated Precision Shader (APS)* architecture is implemented in *Shader Cluster 3*. In addition, the proposed *Approximated Texturing (AT)* mechanism is employed in each texture unit. A texture L2 cache is included for reducing external texture bandwidth. The task buffer is in short for the instruction cache, constant cache, index buffer, vertex attribute and pixel varying buffers. To reach the maximum pixel throughput, the raster engine can interpolate sixteen sets of pixel varyings for shader cores at the same time. At the end of the graphics pipeline, the ROP test unit takes charge of the ROP tests, such as depth and stencil tests. Task dispatcher schedules threads among shader cores, and it realizes the proposed *Screen-space Approximated Lighting (SSAL)* technique by coordinating these shader clusters, the screen-space subdivision test unit in the raster engine and the screen-space reconstruction unit in the ROP engine.

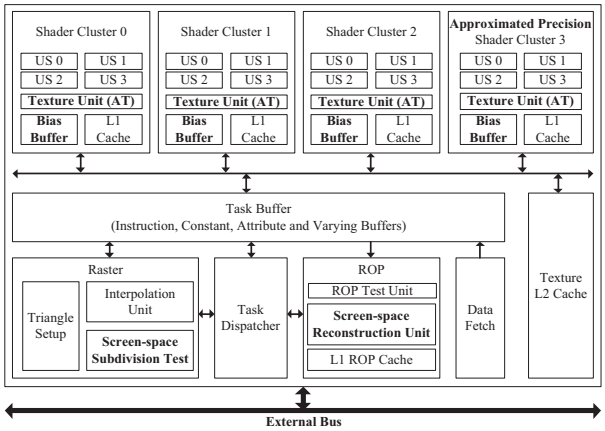


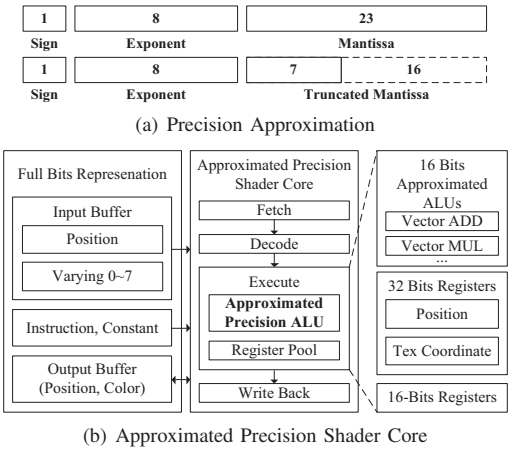
Fig. 1: Proposed GPU architecture.

III. Proposed Techniques

A. Approximated Precision Shader

The *Approximated Precision Shader* architecture is motivated from the heterogeneous multi-core architecture of CPUs in application processors and the precision reduction technique

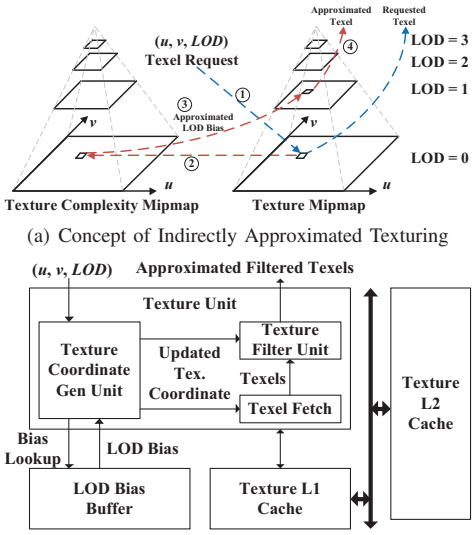
978-1-4799-0280-4/13/$31.00



(a) Precision Approximation

(b) Approximated Precision Shader Core

Fig. 2: *Approximated Precision Shader* architecture.



(a) Concept of Indirectly Approximated Texturing

(b) Modifications in Texture Unit for Approximated Texturing Technique

Fig. 3: Illustration of the proposed *Approximated Texturing* technique.

for shader programming. Heterogeneous multi-core architecture is proved to be power-efficient, where complex cores are employed for heavy-loading tasks to accomplish high performance, and simple cores are employed for lightweight tasks to lower power consumption. On the other hand, Pool *et al.* [4] propose a precision reduction technique in either vertex or pixel shader. In their simulation, substantial power consumption is reduced with satisfactory visual quality.

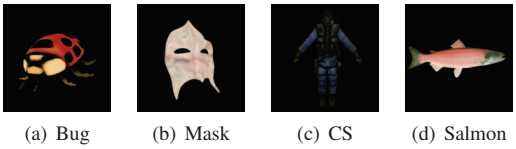
With similar design concepts, we propose an *Approximated Precision Shader* architecture for pixel shader program in the third shader cluster, which reduces the power consumption with slight quality degradation with 16-bit approximated floating-point arithmetics instead of 32-bit, as illustrated in Fig. 2(a). Note that the special 16-bit floating-point format is designed to be easily transformed to/from the standard 32-bit floating-point format. The proposed shader architecture is shown in Fig. 2(b), where the major architectural modification falls on the ALUs and register pool. These vector ALUs are modified as truncated precision representation. Although the corresponding registers can be truncated as 16-bit representation, we reserve the precision of two registers for manipulating positions and texture coordinates since they are more sensitive to precision loss. Therefore, at the beginning of the pixel shader program, we execute move instructions to forward the full precision positions to the output buffer. In our implementation, the area of the approximated precision shader cluster is 61% and the power consumption of executing a pixel program is 55% compared with that of a full-precision shader cluster in average. With an appropriate workload distribution between complex full-precision shader clusters and simple approximated precision shader clusters, low power consumption can be achieved with satisfactory quality.

B. Approximated Texturing

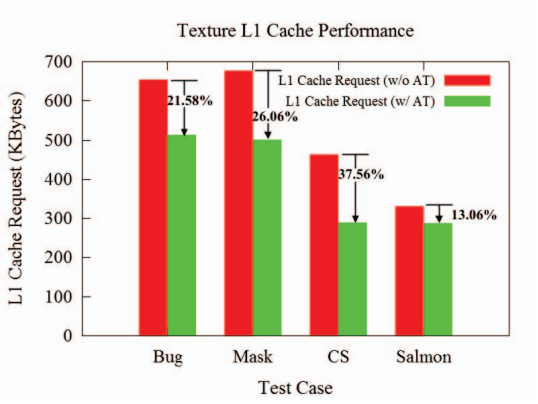
Texture mipmapping is a filtering process for providing correct scale of each rendered fragment during texture magnification or minification. Before rendering, the driver should build a texture pyramid for each level-of-detail (LOD) as shown in the right of Fig. 3(a). A direct texture fetch is illustrated as the blue dotted line in Fig. 3(a) with the rasterized texture coordinates—(u, v, LOD) requested from shaders. For lower texture LOD requests, which contain more texture details and larger memory footprint, it causes enormous switching activities for updating cache blocks, and hence will drain considerable power consumption from L1 caches and L2 caches, moreover, external bus.

The proposed *Approximated Texturing* technique aims to reduce the updating activities of the cache blocks with satisfactory visual quality by accessing smaller memory footprint, that is to say, accessing higher texture LOD. Initially, the texture content is pre-analyzed by wavelet transform to investigate the texture complexity. With the corresponding complexity level, we can adaptively replace the lower LOD texel requests to high LOD when the energy of high bands is low, namely, for flat textures. The concept of approximated texturing is illustrated in Fig. 3(a). As the number indicates the approximation process, there is an extra indirect texture access to the texture complexity mipmap to look-up LOD bias. The bias is quantized as 2-bit in our implementation, which represents that the max LOD bias level is 3. With the encoded LOD bias, the texel request is shifted to higher LOD. In this example, the bias shifts the LOD from 0 to 1, and the required memory bandwidth is thus lowered because of the smaller footprint.

The proposed architecture is shown in Fig. 3(b). The texture coordinate generation unit first looks up LOD bias level from the LOD bias buffer, which stores the LOD bias offset. Compared with a 24-bit (RGB) or 32-bit (RGBA) texture format, the 2-bit LOD bias is relatively small enough to be buffered on-chip. With the LOD bias, the texture coordinate generation unit updates the texture image coordinate and LOD. After that, the texel fetch unit requests texels from L1 cache with the updated texture coordinate. We evaluate the L1 cache performance to verify the reduced cache switching activities and the corresponding approximation visual quality

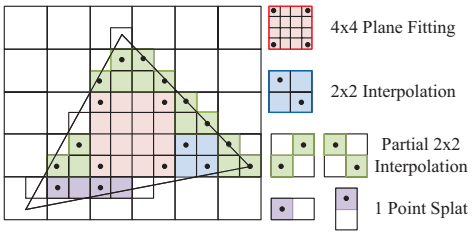


(a) Bug (b) Mask (c) CS (d) Salmon

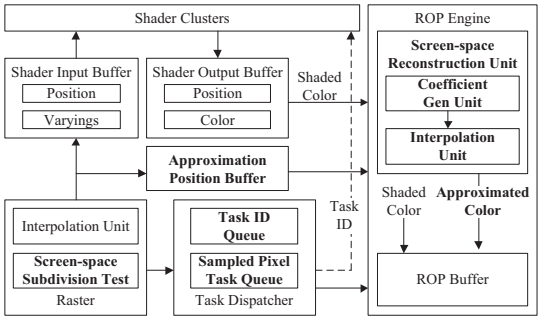


(e) Texture L1 Cache Performance

Fig. 4: The results of *Approximated Texturing*. The corresponding approximated visual quality from (a) to (d) is 29.61 dB, 39.97dB, 36.63dB and 39.55 dB, respectively. (e) Average requested data amount from L1 cache.



(a) Sampling Pattern of Screen-space Subdivision Test



(b) Screen-space Approximated Lighting Architecture

Fig. 5: Proposed *Screen-space Approximated Lighting* technique.

with 512×512 viewports as shown in Fig. 4. Considering the quality degradation caused by the approximated precision shader cluster as well, the average visual quality can reach to 36.4 dB. Fig. 4(e) also shows that the average amount of data that L1 cache requests to L2 cache is reduced by 21.58%, 26.06%, 37.56% and 13.06%, respectively. Our evaluation reveals that 24.57% reduction of L1 cache requests in average is achieved with the proposed technique.

C. Screen-space Approximated Lighting

In [3], Chang *et al.* propose an approximation technique for reducing the switching power dissipation between shader processors and the corresponding buffers while significantly sacrificing the visual quality with annoying blocky effect. In this work, we split the problem domains and further improve the approximation scheme. Textures usually contain most high-frequency components. Straightforward approximating the texturing results in screen-space will seriously ruin the frequency components and degrade the visual quality. Consequently, our proposed *Approximated Texturing* technique solves the problem and shifts the approximation domain to the texture space.

On the other hand, illumination is mainly composed of low-frequency variation on locally smooth object surfaces. The major high-frequency components are caused by geometric silhouette edges. It would be more reasonable to approximate the illuminating operation during a forward lighting pass. Therefore, a *Screen-space Approximated Lighting* technique is proposed. Based on the assumption of locally smooth property, we propose to linearly piecewise approximate or interpolate the un-shaded pixels with the shaded samples. The degree of approximation should be proportional to the projected triangle area, and the concept is illustrated in Fig. 5(a). During a tile-based rasterization process, we hierarchically subdivide the tile to 4×4 screen sub-tiles and 2×2 granularity within it. Different from the previous approximation technique [3], the proposed technique needs to inspect the four corners of a 4×4 sub-tile to assure if it is fully covered. If the four corners are sampled, we can approximate the un-sampled pixels in a 4×4 granularity with a plane equation—$f(x, y) = ax + by + c$. As shown in Fig. 5(a), where the dots denote sampled pixels for shading, each 4×4 approximation can save 75% processor switching activities. If the 4×4 sub-tile check is not passed, the approximation further shifts to 2×2 granularity. For a full 2×2 approximation, it can save 50% of processor switching activities; for a partial 2×2 case, 33.3% is saved. Moreover, for a 1×2 or 2×1 case, we directly splat the intensity of the shaded pixel to the neighboring un-shaded one.

The corresponding architecture is shown in Fig. 5(b). The raster engine employs a screen-space subdivision test unit to inspect these approximation patterns during traversing screen tiles. After the patterns are determined, the raster engine will notify the task dispatcher to issue pixel threads which need to be executed for the shader clusters. The interpolated input variables of the sampled pixels, such as position and varyings, will be directed to the input buffer. In addition, pixel positions identified as approximated pixels are directed to the approximation position buffer. According to the dispatched task ID, shader clusters can request the corresponding pixel input variables. After the sampled pixels are shaded, shader output buffer stores the corresponding pixel positions and colors. Finally, the task dispatcher informs the ROP unit to approximate those un-shaded pixels with the corresponding plane equation. The quality and performance comparisons are shown in Fig. 6. Compared to the technique proposed in [3],

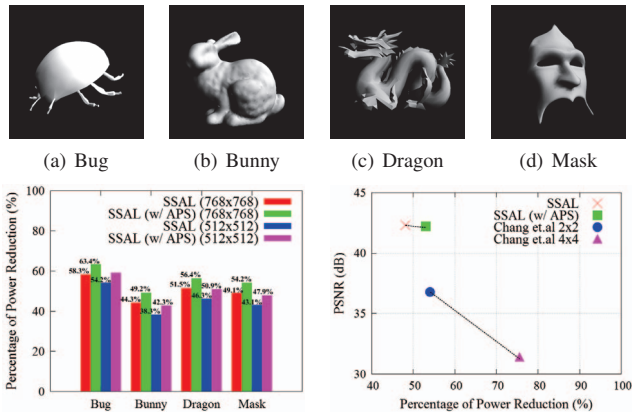


Fig. 6: Performance comparison with the A-SSCC'11 work by Chang *et al.* [3]. The corresponding approximated visual quality from (a) to (d) is 44.14 dB, 42.30 dB, 44.78 dB and 41.43 dB for 768×768 viewport; 43.65 dB, 39.64 dB, 42.21 dB and 39.55 dB for 512×512 viewport, respectively.

Process Technology	**TSMC 45 nm TCBN45GSBWP**
Chip Size	**3.34 x 3.34 (mm²)**
Core Size	**2.8 x 2.8 (mm²)**
Clock Frequency	**350 MHz**
Power Consumption	**130.3 mW**
Arithmetic Performance	**33.6 GFLPOS**
Graphics Performance	**2.8GVertices/s, 5.6GPixel/s**

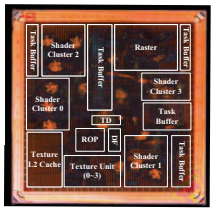


Fig. 7: Chip specifications.

we only average the lighting cases for fair comparison. Our proposed architecture can provide a relative high (over 5.4 dB) and stable visual quality. Furthermore, substantial power reduction compared with [3] can be moderately obtained.

IV. Implementation Results

The chip is fabricated with TSMC 45nm technology. The corresponding chip specifications and micrograph are shown in Fig. 7. The maximum arithmetic operation can reach to 33.6 GFLOPS while the working frequency is 350MHz, and the lowest power consumption is 130.3mW. Considering the rendering performance, 2.8 Gvertices/s and 5.6 Gpixels/s can be achieved. Fig. 8(a) shows the power reduction evaluation of the proposed *Approximated Shader Precision* and *Screen-space Approximated Lighting* techniques, where 11.25% and 47.5% of power consumption can be reduced respectively, and 52.32% of power consumption can be reduced by combining both techniques. Fig. 8(b) shows the comparison between our work and state-of-the-art mobile GPUs [2, 3, 5], where Mvertices/s per mW is adopted as a performance index with process technology normalization for fair comparison. It shows that this work outperforms previous works by at least $1.5\times$. Noted that, for fair comparison, the power consumption of 2 texture units and L2 cache are exclueded from our work, and the comparision between [5] may be not fair because it is not

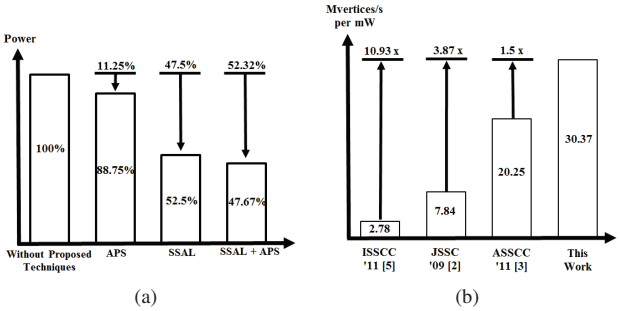


Fig. 8: Power and performance evaluation.

a dedicated design for graphics applications.

V. Conclusion

In this work, we propose a 16-core mobile GPU with three power-aware approximation techniques to improve energy efficiency, which the lowest power consumption is 130mW. The *Approximated Precision Shader* architecture achieve 55% power consumption compared with a full precision shader cluster. In addition, the *Approximated Texturing* technique reduces 24.57% L1 cache requests in our evaluation. Furthermore, the *Screen-space Approximated Lighting* technique can reduce 48% of power consumption, and 53% when combined with the *Approximated Precision Shader* technique. Meanwhile, these proposed approximation techniques can provide satisfactory visual quality.

Acknowledgment

We thank TSMC University Program for supporting the advanced process technology, and Chip Implementation Center (CIC) for EDA tool and measuring services.

References

[1] Y.-M. Tsao, C.-H. Chang, Y.-C. Lin, S.-Y. Chien, and L.-G. Chen, "An 8.6mW 12.5Mvertices/s 800MOPS 8.91mm^2 stream processor core for mobile graphics and video applications," in *Proc. IEEE Symposium on VLSI Circuits*, 2007, pp. 218–219.

[2] B.-G. Nam and H.-J. Yoo, "An embedded stream processor core based on logarithmic arithmetic for a low-power 3-D graphics SoC," *IEEE J. of Solid-State Circuits*, vol. 44, no. 5, pp. 1554–1570, 2009.

[3] C.-M. Chang, Y.-J. Chen, Y.-C. Lu, C.-Y. Lin, L.-G. Chen, and S.-Y. Chien, "A 172.6mW 43.8GFLOPS energy-efficient scalable eight-core 3D graphics processor for mobile multimedia applications," in *Proc. A-SSCC*, 2011, pp. 405–408.

[4] J. Pool, A. Lastra, and M. Singh, "Precision selection for energy-efficient pixel shaders," in *Proc. ACM SIGGRAPH Symposium on High Performance Graphics*, 2011, pp. 159–168.

[5] H.-E. Kim, J.-S. Yoon, K.-D. Hwang, Y.-J. Kim, J.-S. Park, and L.-S. Kim, "A 275mW heterogeneous multimedia processor for IC-stacking on Si-interposer," in *ISSCC Dig. Tech. Papers*, 2011, pp. 128–130.