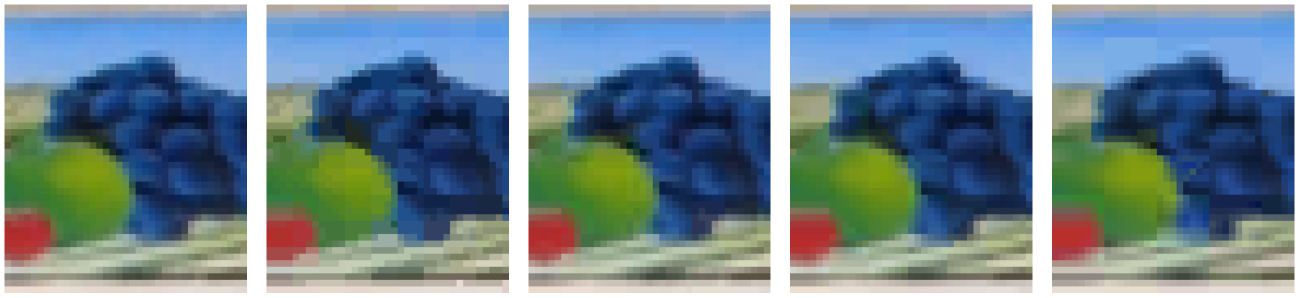
Saving the Planet, One Handset at a Time: Designing Low-Power, Low-Bandwidth GPUs

Thomas J. Olson*
ARM Ltd



(a) *original* (b) *S3TC 4bpp* (c) *ASTC 5.1bpp* (d) *ASTC 3.6bpp* (e) *ASTC 2bpp*

Figure 1: *ASTC Texture Compression gives applications control of the quality vs memory bandwidth tradeoff*

Abstract

GPUs for mobile devices have to deliver ever-increasing performance and capability while living within strict power and memory bandwidth limits. In this talk we'll explore how these limits influence the design of mobile GPUs, and how applications can exploit GPU features to achieve the best power efficiency and performance, using ARM's Mali™ GPU family as a case study.

1 Introduction

Power consumption has become a key factor in determining the performance of modern mobile GPUs. Among the various contributors to GPU power dissipation, external memory bandwidth stands out, because of its magnitude and because it is independent of silicon technology and local circuit optimizations. Consumers of memory bandwidth include frame buffer read/write, texture fetch, display output, and geometry input; which if these is dominant varies with the application, but all are important at one time or another.

To reduce frame buffer read/write bandwidth, many mobile GPUs use some form of deferred rasterization. In this technique, the frame buffer is conceptually divided into fixed-size rectangles or tiles. Triangles submitted by the application are not drawn immediately, but are instead saved in a database that is indexed by the tiles that they overlap. When the application signals that the frame is complete, tiles are rendered one at a time into an on-chip memory store or tile buffer, using the database to extract only the triangles relevant to the current tile. When all triangles for a given tile have been rendered, the tile is written out to the appropriate place in the external frame buffer. Since most depth, stencil, and color buffer accesses are confined to the on-chip tile buffer, frame buffer bandwidth is limited to the final write to external memory. However, application behavior can affect how effective this strategy is, as we'll discuss.

As mobile device screen resolutions continue to grow, even writing tiles to external memory becomes a significant consumer of memory bandwidth. However, in real-world applications it is surprisingly common for many of the tiles in a frame to be unchanged from the previous frame, which makes writing those tiles to memory unnecessary. In the latest ARM Mali GPUs, each tile written to memory is accompanied by a signature. When a newly rendered tile is ready to be written back to memory, the GPU computes its signature and compares it to the one for that tile in the previous frame; if they are the same, the memory write is cancelled. This can save 50% or more of external memory bandwidth in many widely used applications.

With frame buffer bandwidth reduced to a minimum, the next biggest consumer of memory bandwidth for most applications is texture access. While caching helps to minimize redundant texture reads within a frame, most applications have such large working sets that caching texture data between frames is impractical. Instead, application developers are encouraged to use lossy compression methods to reduce texture bandwidth and improve cache efficiency. Unfortunately, existing compressed texture formats lack flexibility and have quality limitations that make them unsuitable for many use cases. To address these problems, we introduce a new format called Adaptive Scalable Texture Compression (ASTC) [Nystad et al. 2012]. ASTC supports bit rates ranging from 8 bits-per-pixel (bpp) down to less than 1bpp in very fine steps, giving the application a high degree of control over the size / quality trade-off. At any bit rate, texels can have from one to four color components. The format supports both HDR and standard (normalized) dynamic ranges, and can encode 3D as well as 2D images. Surprisingly, this flexibility comes with no penalty in quality or coding efficiency; on the contrary, ASTC outperforms S3TC and PVRTC by several dB (PSNR) at comparable bit rates, and is competitive with advanced formats like BC6H and BC7. We'll show examples of the quality at various bit rates, and talk about ways to exploit ASTCs flexibility in game engines and other applications.

References

NYSTAD, J., LASSEN, A., POMIANOWSKI, A., ELLIS, S., AND OLSON, T. 2012. Adaptive scalable texture compression. In *Proceedings of the Conference on High Performance Graphics*, Eurographics Association (forthcoming).

*e-mail:Tom.Olson@arm.com

Copyright is held by the author / owner(s).
SIGGRAPH 2012, Los Angeles, California, August 5 – 9, 2012.
ISBN 978-1-4503-1435-0/12/0008