
SIGGRAPH2015
Xroads of Discovery

The 42nd International Conference and Exhibition
on Computer Graphics and Interactive Techniques

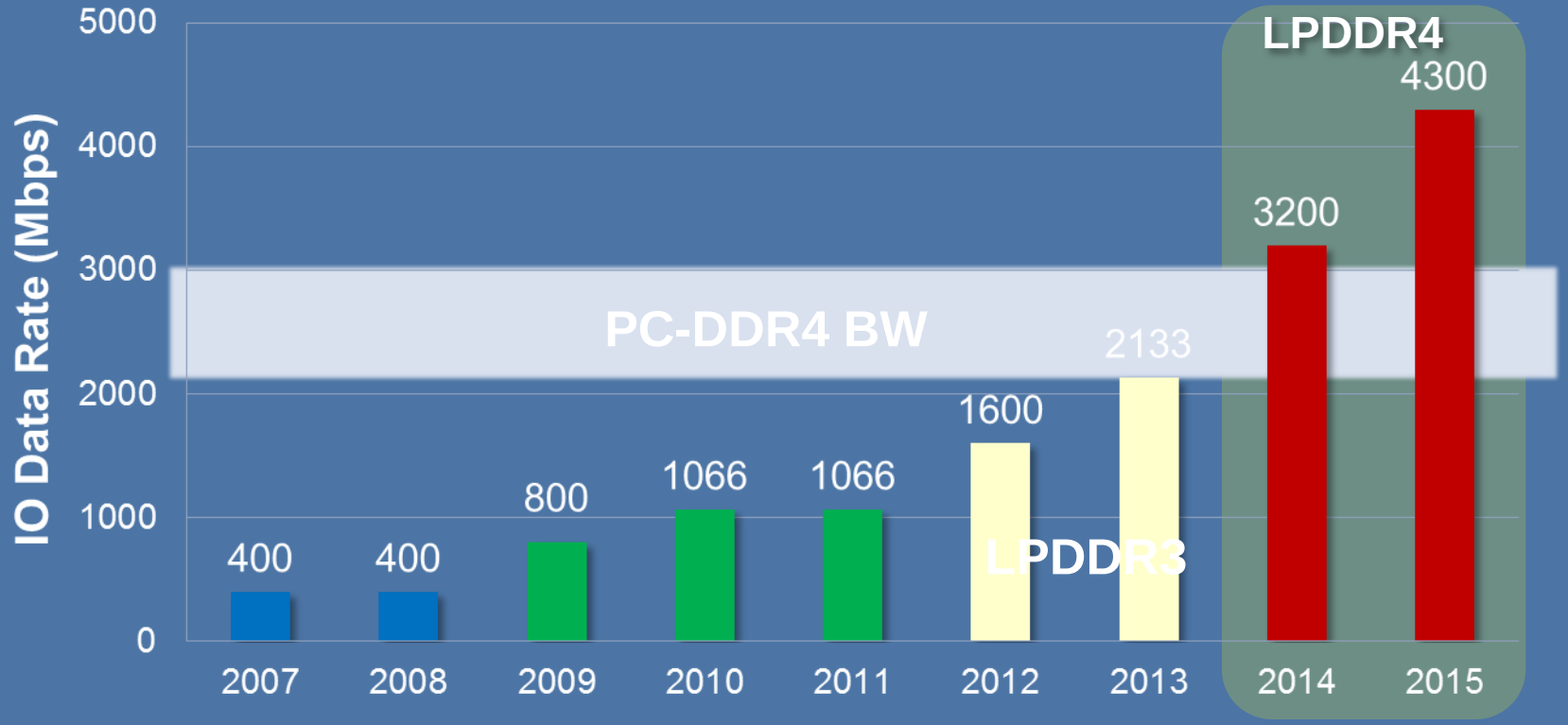
Mobile HW and Bandwidth

Andrew Gruber
Qualcomm Technologies, Inc.

Agenda and Goals

- Describe the Power and Bandwidth challenges facing Mobile Graphics
- Describe some of the Power Saving approaches used in Mobile Graphics to deal with these challenges
  - By no means a systematic account, just a sample of techniques used by major providers
- Provide hints and best practices to allow SW to take advantage of the above approaches.

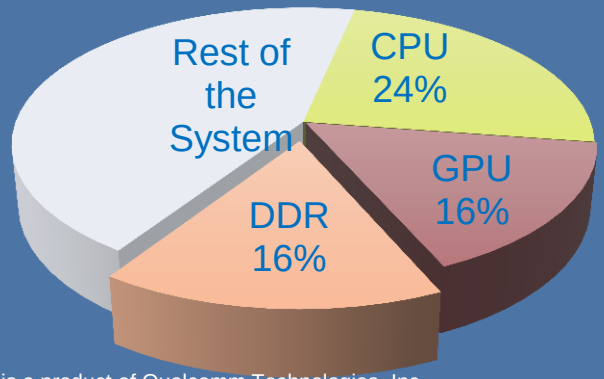
LPDDRx Bandwidth Evolution



Mobile Power Consumption for a Game

- Game running at 1080p / 60 fps
- DDR consumption is ~16% for this gaming use case on Qualcomm ® Snapdragon™ processors.
  - Comparable to internal GPU power and CPU power consumption

Mobile device - Power Breakdown



Typical OpenGL ES 3.1 based game

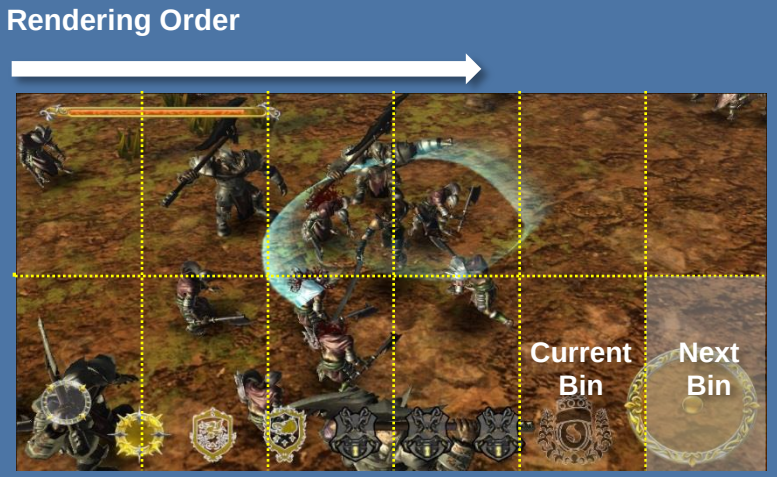
Snapdragon is a product of Qualcomm Technologies, Inc.

Resolutions

- Mobile Resolutions are high and increasing

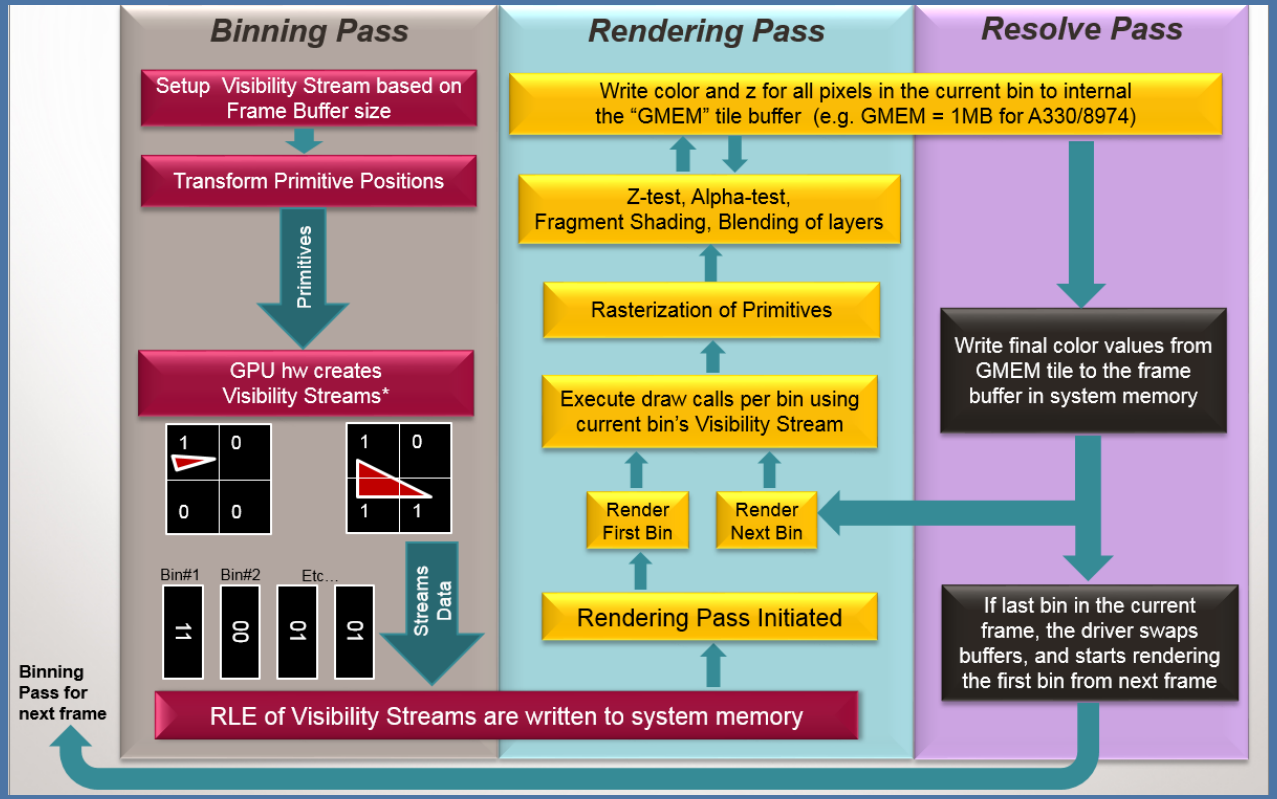
Binning/Tiling Architecture to Save Memory Bandwidth

- GPU has a dedicated fast tile buffer.
- The rendering surface is split up into “bins”.
- Rendering pass draws only pixels that are visible (saving GPU cycles and power)
- The GPU efficiently burst writes all blended pixels from the tile buffer as a single layer to the frame buffer in system memory – referred to as a “Resolve”
- Note that typically there is no Depth traffic to system memory and any MSAA color traffic is first down-filtered before being written to System Memory.



	Opaque	Alpha Blended	Alpha Blended
Tiled	Texture Read	Texture Read	Texture Read, Resolve (Write)
Direct	Texture Read, FB Write	FB Read, Texture Read, FB Write	FB Read, Texture Read, FB Write

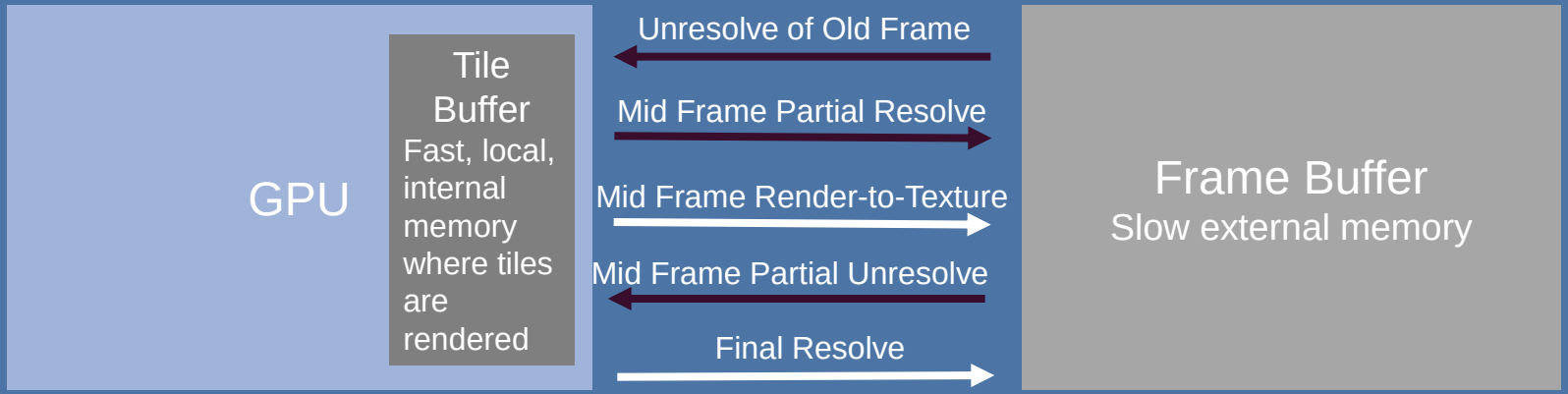
Qualcomm Technologies' Approach to Binning/Tiling



- The only output of Binning Pass is a compressed 1 bit/primitive 'visibility stream'
- Avoids write bandwidth for transformed positions and overflow/
- management of transformed VBO.

Tiling Gotcha's

- Use 'Frame Buffer Discard' to indicate that old data will be overwritten
  - Otherwise we need to copy ('unresolve') the old frame buffer back into the Tile Buffer
  - 'Frame Buffer Clear' works as well – but means we have to waste time clearing the Tile Buffer
- Mid frame calls of glTexImage2D, glBufferData, glReadPixels, glCopyTexImage2D, etc., force the driver to Save/Restore



Bandwidth Saving Technique – Deferred Rending

- ‘Hidden Surface Removal’ or ‘Deferred Rendering’ can avoid rendering occluded pixels – even for non-sorted command streams.
- Works by deferring color render until Z value is known – similar to a Z pre-render.
- Doesn’t work for all cases -- Alpha Blending or Pixel Kill are problematic
- Still Depth sort when possible. Allows ‘Early Z’ to remove a lot of processing

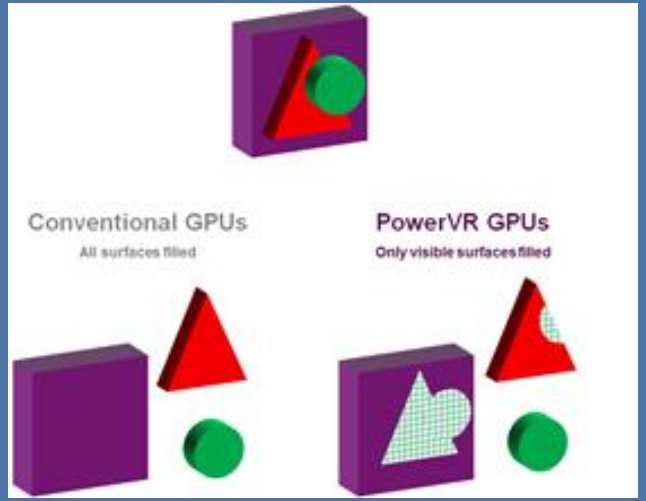


Image used with permission of Imagination Technologies

Bandwidth Saving Technique – Forward Pixel Kill

- Use ‘Early Z’ test to ‘reach into’ processing pipeline and kill pixels that are still being processed.
- Many of the benefits of ‘Deferred Rendering’
  - Kills even with back-to-front Ordering
  - Can kill pixels even if the tile has Alpha Blending.
- Effectiveness drops as tile size or tile complexity increases

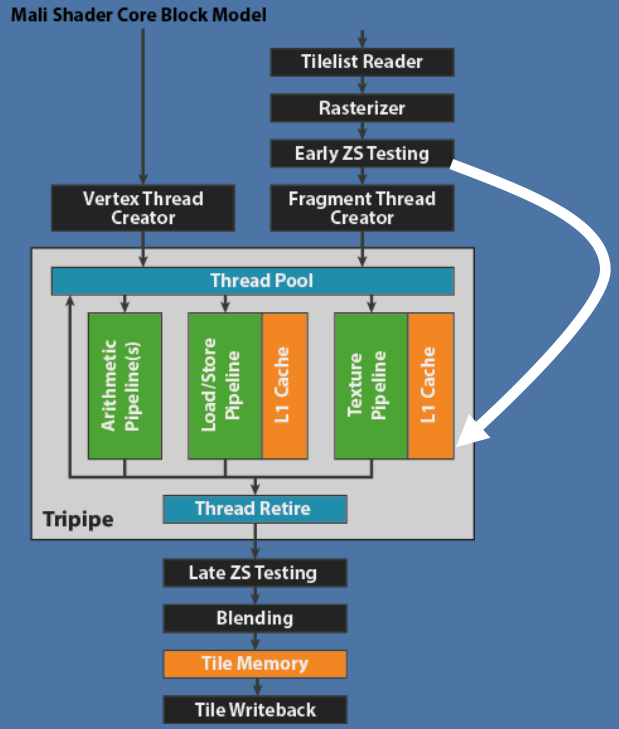
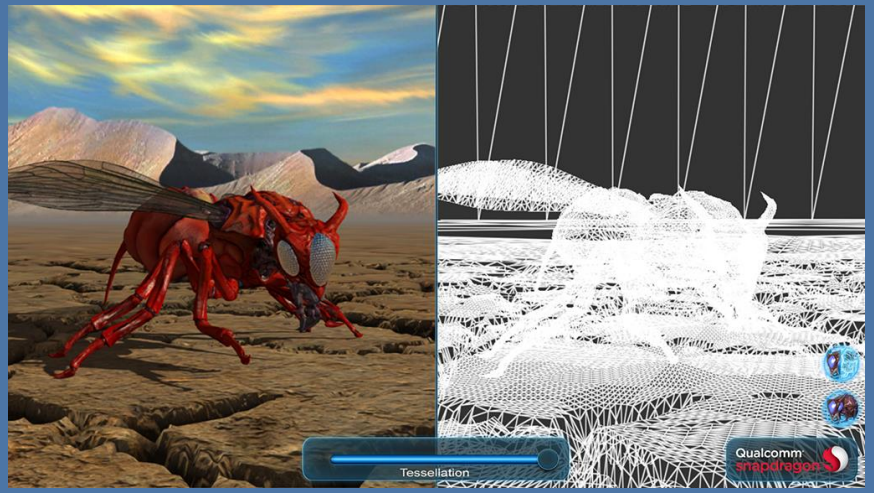


Image used with permission of ARM

Bandwidth Saving Technique – Tessellation

- Supported in AEP/OpenGL ES 3.1
- Provides the ability to render very smooth and high detailed scenes

Tessellation OFF



Tessellation ON

Bandwidth Saving Technique - Tessellation

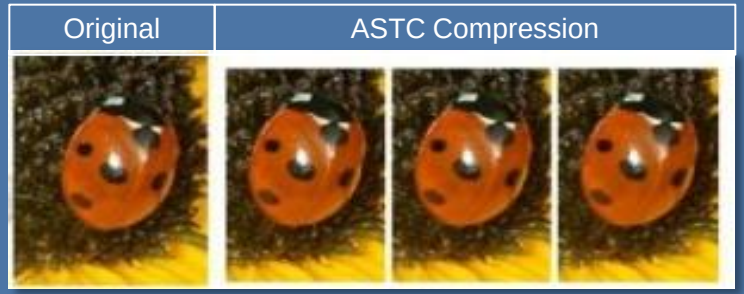
3 Different Possibilities and Bounding Box Extension

1. Tessellate on Binning Pass
   I. Works – but requires extra bandwidth to Save/Read tessellated Vertices
2. Tessellate on Render Pass
   I. Saves bandwidth - but requires tessellated primitive to appear in all bins
   II. Extra tessellation cycles in each Render Pass
3. Tessellate on Both Passes
   I. Limits ‘extra’ Render Pass work for non-visible Bins
   II. But pay extra tessellation cycles during Binning
4. Implementing a ‘Bounding Box’ extension can help limit Render pass work in (2) or Binning Pass work in (3).

Bandwidth Saving Technique - Texture Compression

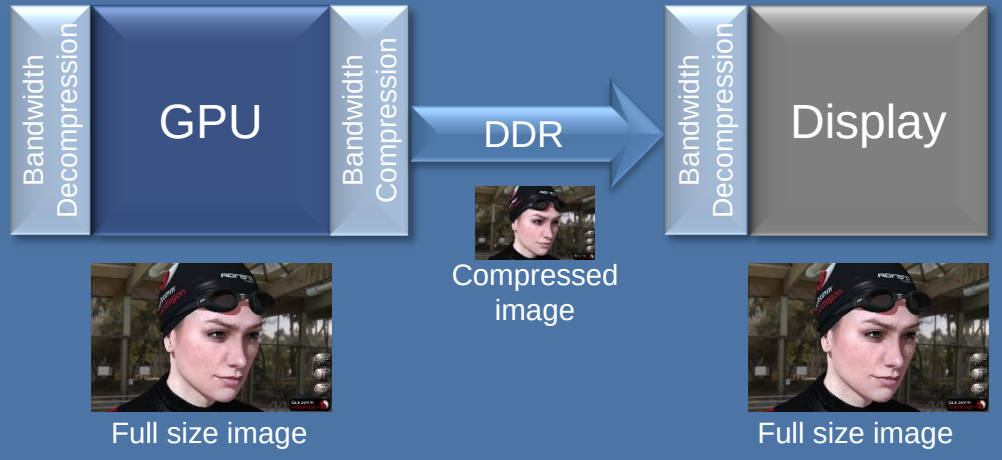
- Texture compression is a critical tool for reducing bandwidth - which improves performance and lowers power consumption
- All textures containing color data should be compressed using one of the recommended formats below.

Target API	Format
Open GL ES 2.0	ATC
Open GL ES 3.0	ETC2
Open GL ES 3.1	ETC2 or ASTC



Bandwidth Saving Technique – Render Target Compression

- Lossless – means that can be invoked anytime the CPU doesn't need access. Doesn't save space -- only memory bandwidth.
- End-to-End (understood by both Texture Unit and Display) means displayable Targets can be compressed – and saves bandwidth on the Scan-out as well.



Summary

- Bandwidth is becoming the dominant performance and power consumption consideration for mobile graphics
- Mobile chips use ‘tricks’ to try minimize bandwidth as much as possible.
- Please cooperate with us by using mobile ‘friendly’ algorithms and using compressed surfaces.