ATI Radeon
HyperZ Technology
Steve Morein
Radeon

- 0.18u CMOS
- 30 M transistors
Radeon Features

- **3D**
  - Charisma Engine
    - Transform, Clipping, and Lighting
    - Key frame Interpolation, multi-matrix skinning
  - Pixel Tapestry
    - 3 texture multi-texture
    - 3D volume texture, 2D cube texture
    - Dot Product per pixel
    - Dependent texture lookup for environment bump mapping

- **Elsewhere**
  - HDTV decoding
  - Adaptive de-interlacing
The Memory Bottleneck

3D is memory bandwidth intensive

- Pixel Operations
  - Texture Read (TR)
  - Z-Buffer Read (RZ)
  - Z-Buffer Write (WZ)
  - Color Read (RC)
  - Color Write (WC)

- Common Color/Z depth is 32-bits (4 bytes)

- Texture Bandwidth
  - Multitexture, Resolution, Texture Compression
  - Net assumption: 32-bits (4 bytes) per pixel
The Memory Bottleneck

Worst case pixel
- \( RZ + WZ + RC + WC + TR = 20 \text{ bytes/pixel} \)
- 40% of bandwidth (8/20 bytes) used for Z data

Common case pixel
- \( RZ + WZ + WC + TR = 16 \text{ bytes/pixel} \)
- 50% of bandwidth (8/16 bytes) used for Z data

Best case pixel (fails Z test)
- \( RZ = 4 \text{ bytes/pixel} \)
- 100% of bandwidth (4/4 bytes) used for Z data
  - But likely to be forced to read texture and color
The Memory Bottleneck

Radeon Fill Rate
- 2 pipes @200MHz = 400 Mpxles/Sec

**Memory Bandwidth Need** (common pixel case)
- 400 Mpxles/Sec * 16 bytes/pixel = 6.4 GBytes/Sec
  - We’ll ignore bandwidth needed to refresh the display

**Available Bandwidth** (common memory system)
- 166MHz, 128-bit DDR SGRAM
- 166MHz * 2 (DDR) * 16 bytes = 5.3 GBytes/Sec
  - Efficiency, of course, is much worse than 100%

**Bandwidth need exceeds available, big problem!**
The Memory Bottleneck

• Typical application today
  – 60% pixels pass z test

• Z is largest user of bandwidth
  – It would be nice to find a way to reduce it

• Overdraw (pixels drawn/pixels per frame)
  – Typically around 3
    • One of every 4 pixels drawn is for clearing the Z buffer

• Not bandwidth limited when drawing pixels that fail Z test
  – But why waste clock cycles to draw hidden pixels?
HyperZ

• Silly marketing name
• What it is:
  – Lossless compression of Z buffer
  – “Fast” Z buffer clear
  – Hierarchical Z buffer
• What it does:
  – Reduce Memory Bandwidth
  – Reduce number of pixels drawn
Z Compression Summary

- Lossless
- Not Application Visible
- Variable Length
  - Block can be uncompressed
    - Required since this is a lossless algorithm
- Reduces Z bandwidth by 50%
Compression Scheme

- 8x8 pixel cache line size
- Can be compressed to:
  - \( \frac{1}{2} \) of original size, “poorly compressed”
  - \( \frac{1}{4} \) of original size, “well compressed”
- Basic algorithm is “DDPCM”
  - Differential differential pulse code modulation
1D Z Compression

8 input z values

Entropy Encoder
2D Z Compression

64 Pixels → 2D DDPCM → Entropy Encoder → Packer
Fast Z Buffer Clear

- Most real-time 3D applications:
  - Clear the Z buffer
  - Do not clear the color buffer
  - Draw all pixels on the screen at least once
- Clear also hurts current PC hardware TCL
Fast Z Buffer Clear

- Avoid clear
- Avoid first read
- A block in memory can be:
  - Compressed
  - Uncompressed
  - Cleared
- Not application visible
Fast Z Clear and Z Compression

<table>
<thead>
<tr>
<th>Application</th>
<th>%Increase in fps (166 Mhz core/166 DDR memory)</th>
</tr>
</thead>
<tbody>
<tr>
<td>3D Winbench (total)</td>
<td>29%</td>
</tr>
<tr>
<td>Quake</td>
<td>24%</td>
</tr>
<tr>
<td>3D Mark (adventure)</td>
<td>24%</td>
</tr>
</tbody>
</table>
Hierarchical Z Buffer

- A Quick 3D pipe review
Hierarchical Z Buffer

Goal
- Remove pixels failing Z test as early in the pipeline as possible
- Remove pixels failing Z test as quickly as possible

Implementation
- Keep reduced resolution Z buffer on chip
- Test pixels early against on chip Z buffer
- Discard pixels before texturing
- Discard at a fast rate (> 8 pixels/clock)
Hierarchical Z Buffer

- 3D pipeline with Hierarchical Z buffer
Hierarchical Z Buffer

- Occluder merging
  - In many cases the occluding object is made of a large number of small triangles, none of which completely occlude the hidden object

- Texture Cache
  - Doing the conservative Z test early prevents the loading of textures used by the hidden object into the texture cache

- “Harder” pixels
  - The pixels that pass the Hierarchical Z test are harder to render; more pass the final Z test.

- Not visible to application
  - Like all of the Hyper Z features, the application does not need to be modified to get a performance boost.
Hierarchical Z Buffer Results

- Application dependent
  - Drawing front to back will optimize performance
  - Some applications already do
  - Benefit even if graphics card does not have Hierarchical Z

<table>
<thead>
<tr>
<th>Application</th>
<th>% pixels fail Z test</th>
<th>% of failing pixels caught by hierarchical Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>3D Winbench (4)</td>
<td>49%</td>
<td>65%</td>
</tr>
<tr>
<td>3D Winbench (9)</td>
<td>24%</td>
<td>93%</td>
</tr>
<tr>
<td>Quake</td>
<td>19%</td>
<td>51%</td>
</tr>
<tr>
<td>3D Mark (cityl)</td>
<td>22%</td>
<td>44%</td>
</tr>
</tbody>
</table>
Future Work

- Some things worked very well
- Some can be further improved
- Extending this to application level culling