Real-World Applications of Computer Arithmetic

Stuart Oberman

Commercial Applications

- General purpose microprocessors with high performance FPUs
  - AMD Athlon
  - Intel P4
  - Intel Itanium

- Application specific processors
  - Digital Signal Processors
  - Graphics Processors

AMD-Athlon Processor Architecture

Raw FP Performance Comparison

<table>
<thead>
<tr>
<th>Operation</th>
<th>Athlon Latency / Throughput</th>
<th>P4 Latency / Throughput</th>
</tr>
</thead>
<tbody>
<tr>
<td>FADD</td>
<td>4/1</td>
<td>5/1</td>
</tr>
<tr>
<td>FMUL</td>
<td>4/1</td>
<td>7/2</td>
</tr>
<tr>
<td>FDIV (SP)</td>
<td>16/13</td>
<td>23/23</td>
</tr>
<tr>
<td>FDIV (DP)</td>
<td>20/17</td>
<td>36/38</td>
</tr>
<tr>
<td>FDIV (EP)</td>
<td>24/21</td>
<td>43/43</td>
</tr>
<tr>
<td>FSORT (SP)</td>
<td>19/16</td>
<td>23/23</td>
</tr>
<tr>
<td>FSORT (DP)</td>
<td>27/25</td>
<td>38/38</td>
</tr>
<tr>
<td>FSORT (EP)</td>
<td>35/32</td>
<td>43/43</td>
</tr>
</tbody>
</table>

AMD Athlon – Newest Offering

- Barton core
- Same basic functionality as original Athlon
- 512KB L2 cache
- 54.3 million transistors
- 74.3W

Microprocessor Performance

DX9 Game: Unreal Tournament 2003
**Microprocessor Performance**

3DMark 2001 SE

**3D Graphics Processing Units: Why?**

- We have software algorithms for “cinematic” quality 3D graphics
- e.g. Shrek, Monsters Inc, Toy Story
- Problem is the rendering time per frame
- Even with larger server farms, can take hours per frame
- Want to achieve same quality in “real-time” on the PC
- Requires 60 fps, instead of 2 hours / frame
- Requires TREMENDOUS amounts of arithmetic computation

**GPUs vs. CPUs**

- More independent calculations
- Enables wide and deep parallelism
- API churn
- Shorter development cycles -> ASIC
- Blend of general- and special-purpose compute resources
- Both transistor-bound for the forseeable future

**Special-Purpose Hardware**

- Most efficient implementations of
  - Cube environment map
  - Shadow calculations
  - Anisotropic filtering
  - Clipping
  - Rasterization
  - Log, exp, dot-product
- More programmability won’t change this

**Recent History: GeForce 1&2**

- First integrated geometry engine & 4 pixels/clk
- Fixed-function transform, lighting, and pixel pipelines
- 25M transistors : 0.18um/6LM : 250MHz
- 25M polygons/sec : 1G pixels/sec

**Rendering in Transition**

- Pre-2001: pixel “painting”
  - Image complexity and richness from LOTS of pixels
  - Each pixel derived from 1-2 textures & blending
  - Detail added by transparency and layers
- Post-2001 fork in the road:
  - Paint more simple pixels, faster - embedded DRAM OR
  - Use Programmable Shading to render “better” pixels - but, must reduce depth complexity
A Tour of the GeForce4

Host / Front End / Vertex Processor
- Protocol and physical interface to PCI/AGP
- Command “ABI” interpreter
- Context switch
- DMA gather

Transform and Lighting
- Handles persistent attributes
- Dispatch
- Hides latency from the programmer
- Fixed-function modes driven by APIs
- Multiple vector floating point processors
  - 256 x 128 context RAM
  - 12 x 128 temp regs
  - 16 x128 input and output

Vertex Program Examples
- Deformation
- Warping
- Procedural Animation
- Lens Effects
- Range-based Fog
- Elevation-based Fog
- Animation
- Morphing
- Interpolation

Primitive Assembly, Setup & Rasterizer
- Per-triangle parameter setup
- Tile walking
- Sample inclusion determination
- Tiles are traversed in memory page friendly order
**Occlusion Culling & Programmable Shading**

- Occlusion Culling reduces Depth Complexity
  - Calculate Z and determine visible pixels
  - Eliminate invisible pixels
- Programmable Shading enables richer visual quality
  - Accurately model: reflections, shadows, materials
  - More textures/pixel
  - More calculations/pixel – consumes many cycles
- Programmable Shading impractical without Occlusion Culling

**Occlusion Strategies**

- Possibilities:
  - Maintain local conservative data structure
  - Use actual depth buffer data
  - Or combine the techniques
  - A coherence problem no matter how you slice it.
- API depth test is at the far end of the pipe!
  - Must preserve semantics

**Pixel Shading / Texturing**

- A pixel shader converts texture coordinates into a color using a shader program.
  - Floating point math
  - Texture lookups
  - Results of previous pixel shaders
- 4 stages, 1 texture address op per stage
  - Compressed, mipmapped 3-D textures
  - True reflective bump mapping
  - True dependent textures (lookup tables)
  - Full 3×3 transform with cubemap or 3-D texture lookup
  - 16-bit-per-component normal maps

**Pixel Shader**

- Input: values interpolated across triangle
- IEEE floating point operations
- Lookup functions using textures
  - Large, multi-dimensional tables
  - Filtered
- Outputs an ARGB value that register combiners can read

**Register Combiners**

- 1–8 stages, plus a final combiner
- Up to 4 inputs from texture stages, interpolators, constant registers, earlier combiners
- Fixed set of operations:
  - Each stage can evaluate \(A \cdot B + C \cdot D\) and output result, along with \(A \cdot B, C \cdot D\)
  - Alternatively, each stage can evaluate dot products instead of multiplies
  - Can conditionally select \(A \cdot B\) or \(C \cdot D\)

**Pixel Shading effects**

- Multi-texturing
- Dot products for per pixel lighting calculations
- Reflections
- Shadowing
- Custom effects
- Pixel math
Texture

- Deeply pipelined cache
- Many hits and misses in flight
- Compression
  - 4:1 ratio
  - Palettes
  - Lossy small-grained fixed ratio scheme
- Filtering
  - Bilinear, trilinear, 8:1 anisotropic

Anisotropic Filtering

Example: Level of Detail Computation

- Texture

- Host / Front End / Vertex Processor
- Frame Buffer Controller
- Transform and Lighting
- Primitive Assembly, Setup & Rasterizer
- Occlusion Culler
- Pixel Shader
- Register Combiners
- Pixel Engines (ROP)

Pixel Engines (ROP)

- Coalesces shader pixels into memory access grain
- Performs visibility and blending / transparency calculations
- Balanced processing power vs. bandwidth
  - Bandwidth is amplified by compression

Statistics

- 136 Mtriangles per second
- 4.8 Gsamples/sec
- 1.2 Tops/sec
- 83.2 GB/sec clear BW
- 63M transistors
- TSMC 0.15u
- 300 MHz pipeline / 325 MHz memory clk
GeForce FX 5800

- Launched Comdex '02
- 125 million transistors
- 200 million Vertices/sec
- DDR-II 500MHz / 1GHz
- 4 billion texels / sec
- First generation of FP shaders, both for geometry and pixel processing (128b)