Graphics Cards and Graphics Processing Units

Ben Johnstone
Russ Martin
November 15, 2011
Contents

- Graphics Processing Units (GPUs)
- Graphics Pipeline
- Architectures
  - 8800-GTX200
  - Fermi
  - Cayman
- Performance Analysis
- Conclusion
Graphics Processing Units

- Processor optimized for graphics processing
- Very good at parallel processing
- Designed for computation-heavy operations
Graphics History

- 1981: First PC video cards created by IBM
- 1984: First processor-based graphics card
- 1990’s: Advent of 3D graphics, advancements driven by gaming/CAD (Computer Aided Design)
- 1992: OpenGL application programming interface introduced
- 1999: Nvidia GeForce 256 marketed as “world’s first GPU”
- 2000’s – present: Increased programmability (DirectX), general purpose usage (GPGPU)
Why GPUs?

- Demand for better graphics (3D)
- Too much work for CPU alone
- 3D pipeline would add complexity to CPU
- **Solution: offload graphics processing to GPU**
**CPU vs. GPU**

- Less data parallelism
- More complex control
- Lower performance goals
- Memory optimized to reduce latency
- Serial programming

**GPU**

- High data parallelism
- Special purpose HW
- Memory optimized for high throughput
- Less emphasis on cache
- Stream programming
Characteristics

- Stream processing
- Lots of functional units
- Several cores (>100)
- Multiple, light threads
- Fast memory
- 2 types of programmable processors: vertex and fragment
Instructions

- Computation-heavy, few memory accesses (has what it needs beforehand)
- Independent and highly data parallel (same instruction performed on several data elements)
Stream Computation/Graphics Pipeline

- Stream: ordered set of data of the same type
- Efficiency increases with stream length
- Kernel: operation performed on an entire stream
- Stream applications are created by stringing kernels together
- Creates a “graphics pipeline”
Inside the Pipeline

Vectors (from CPU)

(x,y,z,w)
(r,g,b,alpha)

Texture Cache

Primitives

Points, lines, triangles

Fragment Processor

Pixels

*Final color

Rasterizer

Pixels
The Vertex Processor

- Polygonal models are made up of vertices
  - (x,y,z,w) coordinates
  - (r,g,b, \( \alpha \)) component colors
- Vertex processor applies a vertex program
- Changes position of vertices
- Makes use of texture cache (modern GPUs only)
- Output of vertex processor goes to the rasterizer
- Rasterizer converts primitives into fragments
The Fragment Processor

- AKA: Pixel Shader
- Fragment: contains all information required to make a pixel
- Fragment processor performs fragment program
- Computes final color of pixel
- Utilizes data from texture cache
General Architecture

Fatahalian 2009
Geforce 8800-GTX200 Architecture

• Very Close to Tesla Architecture
• 16-30 SMP Cores Running up to 768 Threads Each
  • Warps of 32 Threads
• Each Core
• Unified Pixel/Vertex Shader Architecture
• 6 Pairs of L2 Cache and Memory Buses
  • L2 Cache is Read-Only for Textures
• First card with DirectX 10 Support
Geforce 8800

Owens 2007
Tesla Architecture – GTX200

Luebke 2008
Fermi Architecture

- 40nm Process, 3 Billion Transistors
- 512 CUDA cores organized into 16 SMPs
  - Each supporting up to 1536 simultaneous threads
  - 16 Load/Store, 16 ALU, 4 Special Purpose Units Each
- 6 GDDR5 DRAM Interfaces, 40 bit Addresses
- Cores Clocked at 1.3Ghz+, while Cache is at 600Mhz
Fermi Architecture

- Embedded Unified Shader Architecture
  - Incorporates Tessellation into Vertex, Geometry, and Pixel Shading
- 8x Double Precision FP Performance over Tesla
- Distributed Rasterizer
- L1 Cache - Register Spilling, Stack Operations, and Load/Store only
- L2 Cache – For Textures, Read/Write
- ECC Memory Protection
Fermi Architecture

Nickolls and Dally 2010
AMD’s Cayman Architecture

- 40nm Technology, 2.64 Billion Transistors
- 24 880Mhz Cores
  - Each Running 4-Wide VLIW Instructions
    - Static Scheduling
- 8KB L1 Texture Cache
- Shared L2 Cache
- 4KB Write Cache
- OpenCL and DirectCompute instead of CUDA
- Interleaved ALU Functions
  - All ALU/Execution Units are same, no special units
- 7840KB of Registers
AMD’s Cayman Architecture

## Architecture Comparisons

<table>
<thead>
<tr>
<th>Card</th>
<th>Stream Processors</th>
<th>Graphics Clock Speed</th>
<th>Peak Memory Bandwidth</th>
<th>Power</th>
<th>Peak GFLOPs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Geforce 8800GT</td>
<td>112</td>
<td>600Mhz</td>
<td>57.6 GB/s</td>
<td>105 W</td>
<td>504</td>
</tr>
<tr>
<td>GTX 280</td>
<td>240</td>
<td>602Mhz</td>
<td>141.7 GB/s</td>
<td>236 W</td>
<td>933</td>
</tr>
<tr>
<td>GTX 480 (Fermi)</td>
<td>480</td>
<td>700Mhz</td>
<td>177.4 GB/s</td>
<td>250 W</td>
<td>1344.96</td>
</tr>
<tr>
<td>Radeon HD 6950 (Cayman)</td>
<td>960</td>
<td>800Mhz</td>
<td>160 GB/s</td>
<td>200 W</td>
<td>2253</td>
</tr>
</tbody>
</table>
Performance Metrics

- Based on Warp Parallelism
- Memory Warp Parallelism
  - How many concurrent memory requests
  - # Warps Accessing Memory in each Memory Period
- Computational Warp Parallelism
  - How much computation can be done in warps while one waits for memory
  - CWP = (Memory Cycles + Comp Cycles)/Comp Cycles
- Parallel Element Scalability
  - = # Elements/Total Execution Time
- Parallel Subset Scalability
  - = (Execution time of Subset) * (# Subsets / # GPU Cores)
GPU Performance Benefits

- Jacobi Algorithm – 2-7.5x Speedup
- Monte Carlo Simulations – 16x Speedup
- EDA Scheduability – 17x for Entire Problem, 100x in Kernel Computations
- Euler Algorithm – Up to 500x Speedup
  - Equivalent to 125 Quad-Core CPUs
Future Architectures

- Nvidia’s Kepler
  - 28nm Technology
  - Supposedly in 2011, but Likely not until Mid 2012
  - DirectX 12 Support
  - 3-4x Performance over comparable Fermi cards, with 3-4x less power consumption
  - Support for Pre-emption and virtual memory

- Nvidia’s Maxwell
  - Late 2013

- Nvidia’s Echelon
  - 2017
  - 10nm Technology
Future Applications/Obstacles

- General Purpose GPU (GPGPU)
  - Simulation, video encoding, numerical methods, databases
  - GPU used to process gaming physics/artificial intelligence?
- More programmability
- Power management
- Memory bandwidth limits
  - Faster PCI-Express buses
  - Combined CPU/GPU
Questions?

So what do we do if video game AI opponents become smart enough to question the "Matrix" into which we've put them?

Wait a minute! None of this is real! I can see through the world! I can see the code! I am the one!