Overview of Nvidia GeForce 6 Series Architecture and More

Prepared by: Dustin Balise
Overall System Architecture

Block Diagram

Memory Hierarchy

Graphics Pipeline

- Programmable Vertex engine
- Programmable fragment engine
- Texture load/filter engine
- Depth-compare and blending data write engine

Graphics Pipeline for Non-Graphics Operations

- Vertex and Fragment processor are highly computationally capable
- Texture unit used as random-access data fetch unit
  - 35 GB/sec

CPU-GPU Analogies

- GPUTextures = CPU Arrays
- GPU Fragment Programs = CPU “Inner Loops”
- Render-to-Texture = Feedback
- Geometry Rasterization = Computation Invocation
CPU-GPU Analogies

- Texture Coordinates = Computational Domain
- Vertex Coordinates = Computational Range
Performance

- 425 MHz graphics clock
- 550 MHz memory clock
- Vertex Processor
  - 6 four-wide fp32 vector MADs per clock cycle
  - One scalar multifunction operation (such as sine or reciprocal square root) per clock cycle
Performance

- Fragment Processor
  - 16 four-wide fp32 vector MADs per clock cycle
  - 16 four-wide fp32 multiplies per clock cycle
Branching

- Fragment Processor works on many fragments at the same time
  - Fragments in group may take different branch
  - Fragment Processor needs to take both branches
  - 6 cycle overhead for if-else-endif control structures
That was in 2005…

- Geforce 8 series
  - 450-675 MHz core clock speeds
  - 400-1080 MHz memory clock speeds
  - 256-768 MB of memory
  - 6.4-103.7 GB/s memory bandwidth
  - Costs range from about $150-$700
Diagram of High End Nvidia GPU

HPC Solutions

- Tesla C870
  - 128 multi-threaded processors per GPU
    - Full integer and floating point operations
  - C-language development environment and a suite of developer tools (CUDA)
  - 1.5 GB of Dedicated GDDR3 Memory
  - Over 500 gigaflops of peak floating point performance
  - 76.8 GB/s Memory Bandwidth
  - Parallel data cache
CUDA

• Nvidia SDK for general purpose computing on GPU’s (GPGPU)
• Compatible with Nvidia 8 series, Quadro FX 4600/5600, and Tesla GPU’s
• Runs on Linux and Windows
Cuda Source Files

- **Host Code**
  - Runs on generic x86 processor
  - C and C++ source files

- **Device Code**
  - Runs on GPU
  - “C like” source file
  - Basically GPU functions
CUDA Compiler (nvcc)

● Separates device functions from host code
● Passes host code to platform compiler (i.e. gcc, g++ …)
● Embeds compiled GPU functions as load images in the host object file
● Linking stage provides support for remote SIMD procedure calling and explicit GPU manipulation
Bibliography

