Panel: Fixed-Function Hardware

What role will it play in future graphics architectures?

Fixed-Function Hardware

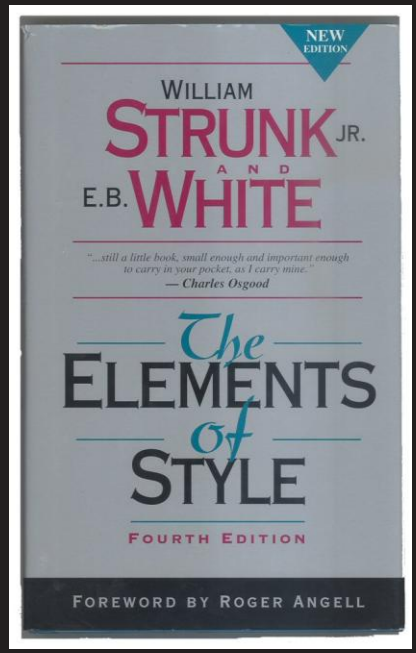
- GPU programmers love to hate it
- Its performance enables interactive graphics
- Its power efficiency enables mobile computing
- Mobile phones suggest a ubiquitous future
- And even CPUs are full of it

Panelists

- Kayvon Fatahalian, Stanford
- Mike Houston, AMD
- David Blythe, Intel
- Steve Molnar, NVIDIA
- Johan Andersson, DICE

Guidelines

- “Make definite assertions”
- “Prefer the specific to the general”
- “If you don’t know how to pronounce a word, say it loud!”



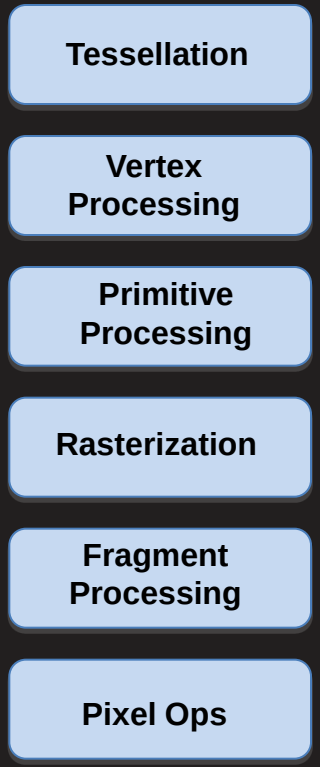
Kayvon Fatahalian
Stanford

Different types of fixed-function

Compute-intensive tasks

- Texture filtering
- Transcendentals (sin, cos)

- Rasterization



“Irregular” tasks
(aka. compute-organizing tasks)

- Filtering: early zcull
- Packing: fragments into warps
- Grouping: vertices into prims
- Managing access to queues
- Distributing pipeline work
- HiZ build/update

My question

- What is the future of the fixed-function stuff that handles irregular tasks and “organizes” computation on GPUs?

- If it remains
  - Does it generalize to assist a wider domain of apps?
  - If so, how does it get abstracted to software?

Beyond Programmable Shading Course
ACM SIGGRAPH 2010

Vive la Fixed Function!

Steve Molnar
NVIDIA

Overview

- Simple Analysis
- The Flaw in the Simple Analysis
- Power Changes Everything

Simple Analysis

- Assume two ways of performing an operation
  - Fixed-function hardware
    - Performs operation with efficiency E_f (ops/mm^2/sec)
    - Hardware is unused otherwise
  - General-purpose hardware
    - Performs operation with efficiency E_g
    - Can perform any other operation
    - Perfectly load-balanced and fully utilized

Which is better?

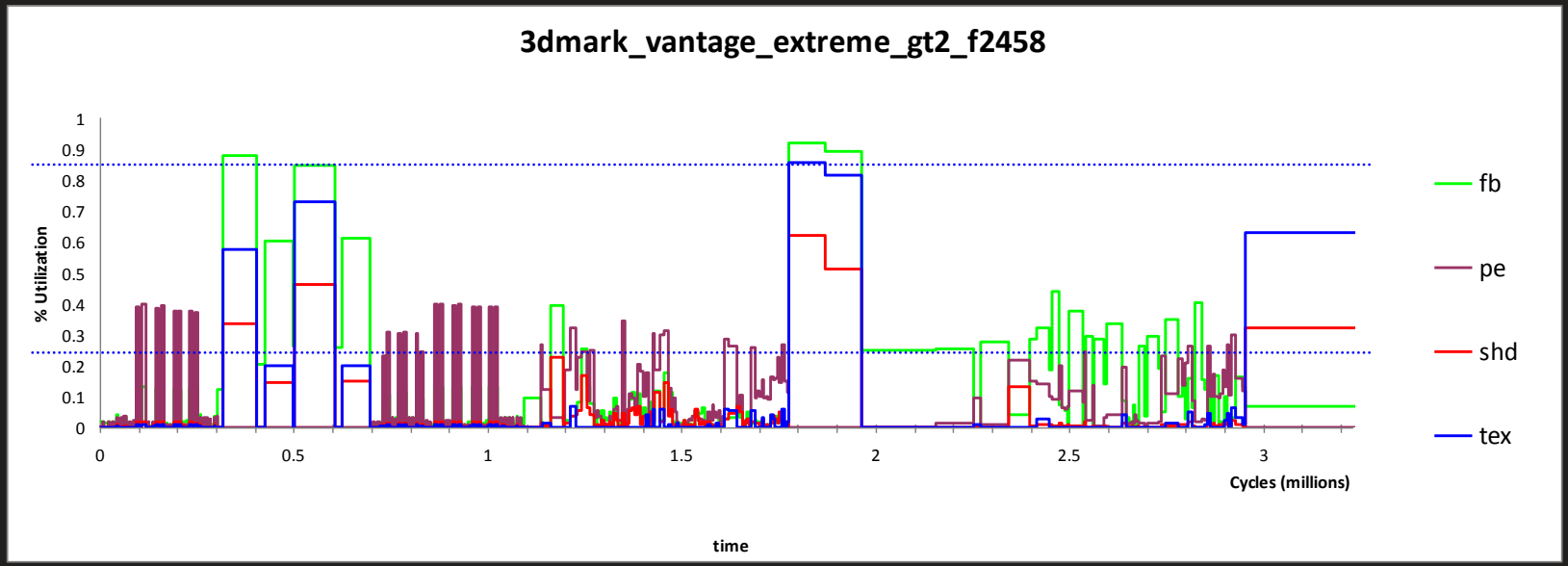
- Assume operation is fraction μ of overall task
- Fixed function wins if $\mu E_f > E_g$
  - E.g. if bilinear texture filtering hardware is 4x denser than filtering in fragment shader, fixed-function texture wins if utilized more than 25% of the time

The flaw in the simple analysis

- Most tasks we care about have variable load

Peak tex util →

Avg tex util →



If you undersize a fixed-function resource…

- If you undersize a fixed-function resource relative to the instantaneous load, every other system resource (including your expensive shader) is under-utilized.
- Don’t do this! (at least don’t do it much)
- So you have to oversize fixed-function resources
  - Design for near-peak utilization μ′
  - Now fixed function wins if $\mu \cdot (\mu/\mu') \cdot E_f > E_g$
  - (μ/μ') is often << 1, greatly reducing appeal of fixed function

Power Changes Everything

- Systems of nearly every size are becoming power-constrained
  - Mobile systems: battery life is crucial
  - High-end systems: limited by power supply, cooling, noise
  - Market rewards efficient, quiet systems

Power depends on area

- Two types of power:
  - Static (leakage) power: proportional to circuit area
  - Dynamic (active) power: proportional to circuit area * fraction of gates that switch
- A smaller fixed function unit has lower static and dynamic power
  - Example from Billy Dally:
    - Picojoule to do 32 bit multiply
    - Nanojoule to do 32 bit multiply on a CPU (read, write cache, etc.)
  - Power really favors fixed-function

Mobile parts are harbinger of future

- Example: NVIDIA Tegra 2
  - System on chip with:
    - ARM A9 dual-core processor *and* ARM A7 (low-power) processor
    - 3D core
    - MPEG encoder/decoder
    - MP3 decoder
    - Image signal processor
  - Only active cores powered on
    - 140 hours mp3 play
    - 12 hours 1080p video decode
    - 6-8 hours 3D

Vive la Fixed Function!

- As long as key parts of mainstream applications can be implemented more efficiently in fixed-function, power argues that they should be
- Examples:
  - Texture filtering
  - Sum-of-products arithmetic
  - Compression/decompression (texture, depth, color)
  - Arguable: pipeline sequencing
- Challenge to architects of future systems: refactor architecture so core algorithms currently implemented in fixed function can be implemented with same efficiency in programmable framework

Backup

Economics of area and power

- Transistors per chip are still increasing at close to Moore's Law
- Power per device used to follow same curve
  - Because of leakage, now on shallower curve