Graphics Papers

Lecture #14: Tuesday, 21st May 2002
Lecturer: Prof. Bill Dally
Scribe: Timothy Knight, Jung Ho Ahn
Reviewer: Mattan Erez

1 Comparison of Different Architectural Approaches

The following table(s) contain description of the techniques used to handle parallelism, the hiding of memory latency, gaining sufficient memory bandwidth, and design and programming complexity for the 3 different architectural approaches discussed in class.

	Vectors	Multithreading	Streams
Parallelism	Data level parallelism at the operation level.	Thread level parallelism. • Synchronization costs are high. • Instruction cost - instruction fetches per computation. • More flexible (MIMD vs. SIMD)	Data level parallelism at the kernel level. • Gives better reuse than data parallelism at the operation level.

Table 1: Comparison of Different Architectural Approaches

	Vectors	Multithreading	Streams
Memory Latency	Hides latency by: • Overlapping memory access with computation. • Amortizing the latency cost over the length of the vector.	Hides latency by: • Switching threads - do something else while waiting. • Replicating state to allow fast context switching. Enables data-dependent pointer chasing, since with vectors and streams need to know ahead of time the memory addresses needed in order to amortize latency.	Hides latency by: • Overlapping memory access with computation. • Amortizing the latency cost over the length of the stream. • Exploiting producer-consumer locality.
Bandwidth	Need a very high bandwidth memory system. No advantages over conventional processors.	Threads compete for the cache. • Can get interference, or synergistic sharing.	Reduces demand with producer-consumer locality. Can get some of this advantage in a conventional machine using a cache, but a stream machine has a better bandwidth heirarchy, illustrated in figure 1.

Table 2: Comparison of Different Architectural Approaches (cont.)

	Vectors	Multithreading	Streams
Design Complexity	Simplest to design.	High complexity: • Multiple instruction. • Synchronization. • etc.	Moderate complexity: • Microcontroller. • LRFs.
Program. Complexity	Easy - vector ops.	Familiar but difficult: • Synchronization. • 'Thread-safe' code.	Need to make some things explicit, such as global references.

Table 3: Comparison of Different Architectural Approaches (cont.)

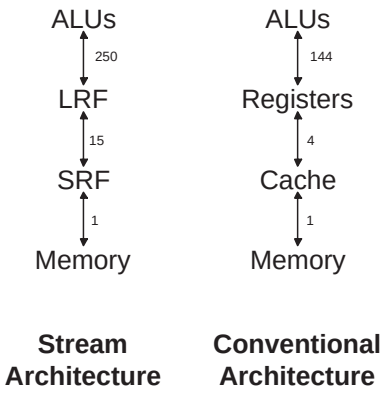

Figure 1: Bandwidth Hierarchies: Imagine vs. Pentium-4

2 NVIDIA Graphics Paper

The NVIDIA paper 'A User-Programmable Vertex Engine' by Lindholm et. al. was discussed.

- It presents an introduction of stream processing into a fixed function pipeline.
- They removed a fixed stage and replaced it with a 'stream processor' without a memory system - basically, a cluster.
- The question was raised: Are graphics chips becoming stream processors? Due to being in a price sensitive market, they will most likely never become truly general purpose, rather become multithreaded SIMD processor. But they are trending in that direction.
- Will a graphics processor ever be used to do general purpose computation? Maybe for niche applications; difficulties with making random memory accesses. More likely to have a processor extension which is stream-like - need to standardize a platform so that software vendors can target it.

3 Purcell Graphics Paper

The paper 'Ray Tracing on Programmable Graphics Hardware' by Purcell et. al. was discussed.

- They're doing random memory accesses through the texture cache.
- They brought up the issue of multipath vs conditional loop. They want branching for conditional loops and not for arbitrary conditionals in order to achieve the efficiency.
- They made a very high level simulator of a new (non-existing) graphics chip to perform their experiments. This is a pretty efficient way to work, because no processor exists right now which satisfies their architectural necessities. They took performance values from a current NVIDIA chip and extrapolated those for what they wanted.
- They need an SRF? There exists locality, so if they could keep the state of rays around, less memory bandwidth would be needed.
- GPU (graphic processing units) are becoming more general, but are still essentially dedicated graphics chips.

4 Other Comments

- For general storage units, the bigger the slower.
- Some pitfalls in Imagine which limit its use as a graphics processor, specifically it has fewer arithmetic units per unit area than specialized graphics chips.
- Pin packaging trends and issues were introduced - number of pins on a package is increasing much more slowly than number of transistors in a chip. Pins can be time-multiplexed at a very high frequency to attain a greater effective bandwidth into and out of the chip.