EE 7722—GPU Microarchitecture

URL: `http://www.ece.lsu.edu/gp/`.

Offered by:

David M. Koppelman

3191 P. F. Taylor Hall, 578-5482, koppel@ece.lsu.edu, http://www.ece.lsu.edu/koppel

Office Hours: Monday - Friday, 14:00-15:00.

Prerequisites By Topic:

- Computer architecture.

- C++ and machine language programming.

Text

Papers, technical reports, etc. (Available online.)

Course Objectives

- Understand low-level *accelerator* (including GPU and many-core) organization.

- Be able to fine-tune accelerator codes based on this low-level knowledge.

- Understand issues and ideas being discussed for the next generation of accelerators.

Course Topics

- Parallelism Fundamentals
- GPU Architecture and CUDA Programming
- GPU Microarchitecture (Low-Level Organization)
- GPU Algorithms
- Tuning based on machine instruction analysis.
- Many-Core Accelerator Xeon Phi Architecture and Microarchitecture

Graded Material

Midterm Exam, 35%

Fifty minutes in-class or take-home.

Final Exam, 35%

Two hours in-class or take-home.

Yes, it's cumulative.

Homework, 30%

Written and computer assignments.

Lowest grade or unsubmitted assignment dropped.

Course Usefulness

Material in Course Needed For:

- Those designing the next generation of accelerator chips.

- Those writing high-performance scientific programs.

- Those writing high-performance graphics programs.

- Those interested in future computer architectures.

- Compiler writers.

Course Resources

- Slides and other material via `http://www.ece.lsu.edu/gp/`

- Code examples in Git repository `git://svn.ece.lsu.edu/gp`

- Web site also has homework assignments, exams, grades, and other material.

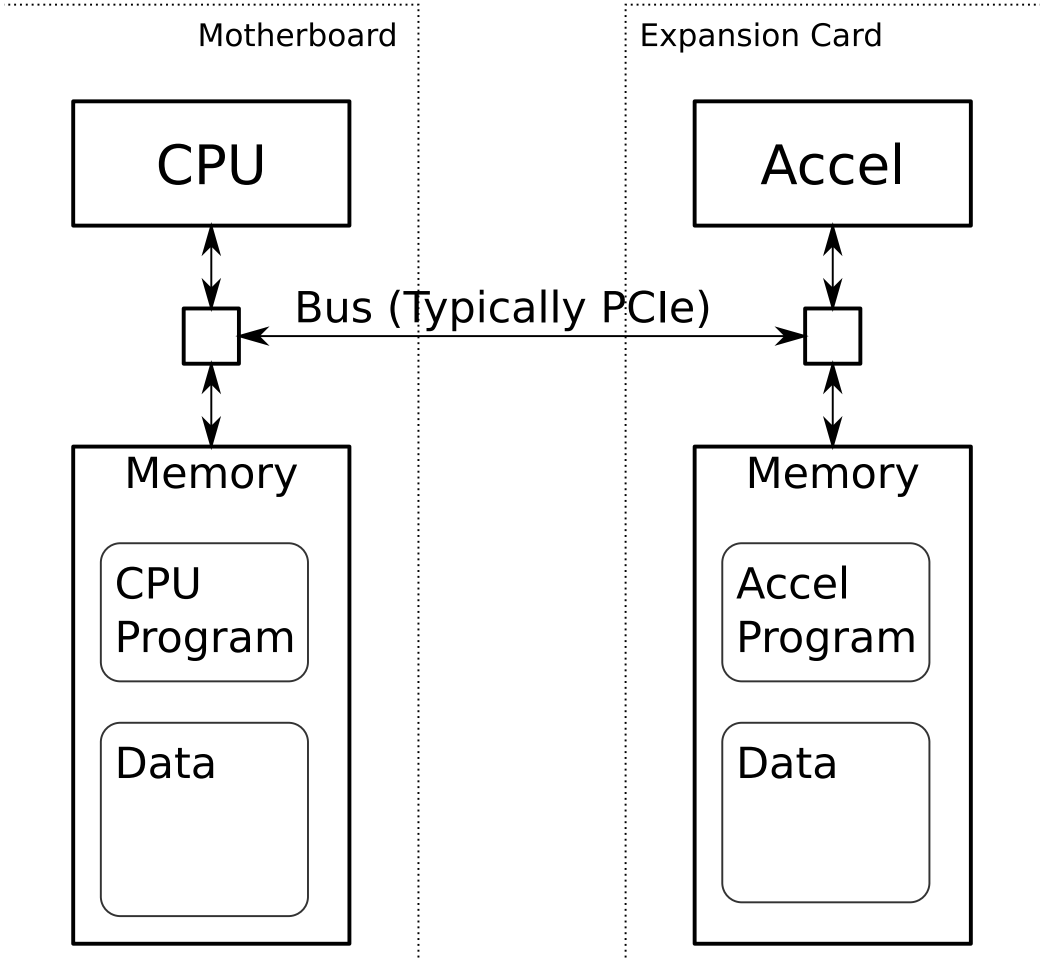
- Announcements are on course home page and available as a Web (RSS) Feed.

Accelerator:

A specialized processor designed to work alongside a general-purpose processor (CPU).

Work is split between the different devices . . .

. . . each does what it's best at.



Common Accelerator Types

- *GPU (Graphics Processing Unit)* — *E.g.*, NVIDIA Kepler K20
- *Many-Core CPU* — *E.g.*, Intel Xeon Phi
- *FPGA Accelerator* — *E.g.*, Nallatech PCIe-180

Example:

> “Our computer was taking four days to compute the 48-hour forecast, so we bought a system with 3 accelerators: an NVIDIA K20c, a Xeon Phi, and a Nallatech board, all of these were given work that would have been performed by the general-purpose CPU, an Intel i7.”

GPU (Graphics Processing Unit):
A processor designed to execute a class of programs that includes 3D graphics and scientific computation using a large number of threads.

A Brief History

GPUs originally designed *only* for 3D graphics.

Large economies of scale made them cheap.

Resourceful scientific users disguised their work as 3D graphics.

GPU makers started supporting scientific and other non-graphical work.

GPU evolved into a second kind of processor, with 3D graphics just one application.

GPU Product Examples

- NVIDIA GTX 780 — High-end GPU for home use.
- NVIDIA Kepler K20x — High-end GPU for non-graphical computing..
- AMD Radeon R9 — High-end GPU for home use.

Many-Core Processor:

A processor designed to execute a class of programs that includes 3D graphics and scientific computation using simple cores and wide vector units.

A Brief History

Long known that peak performance of chip of small cores > chip of large cores ...
... the problem was parallelization.

Many research and one-off designed used chips filled with simple cores.

The inclusion of wide vector units meant few cores would be needed.

Idea used by Intel for a graphics chip, project Larrabbe.

Larrabbe re-targeted at scientific computing, product named Phi.

Many-Core Processor Examples

- Intel Xeon Phi — For scientific computing.
- Sun T2000 — Meant for server workloads.

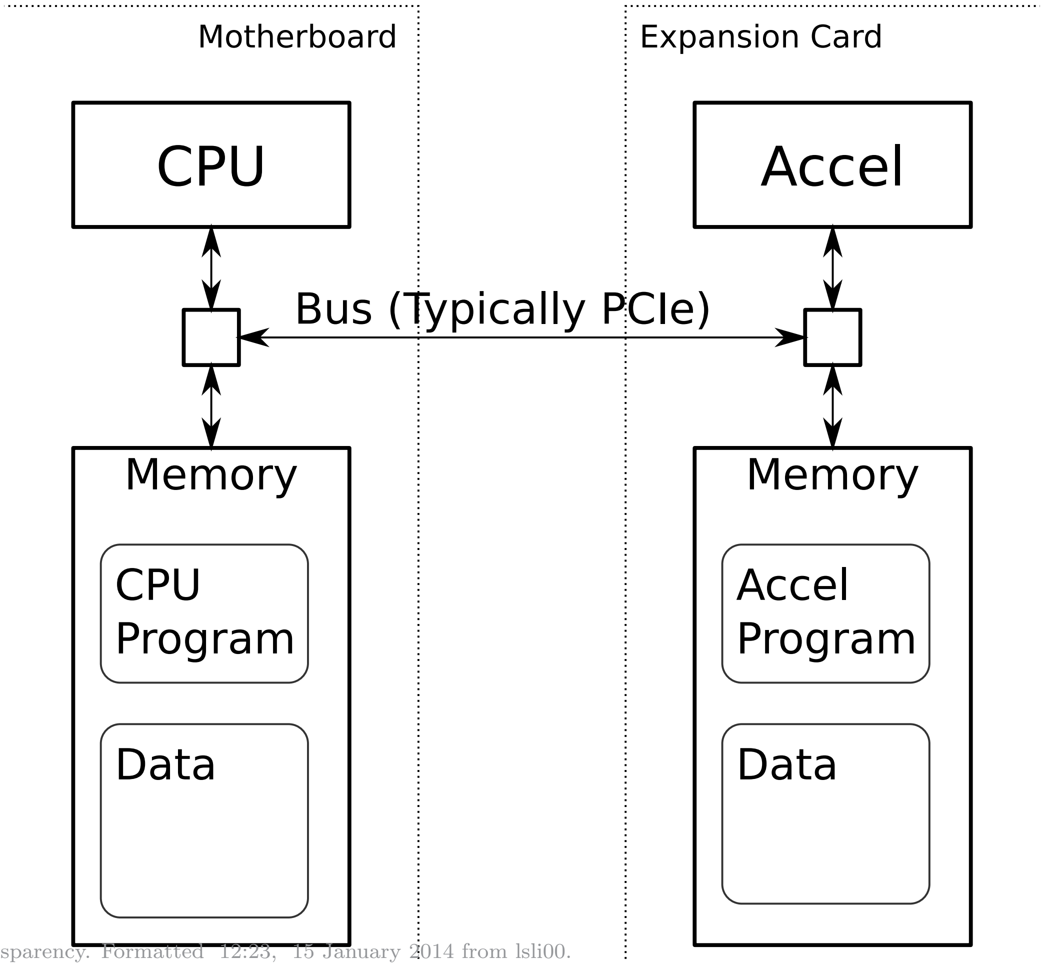
Organization of Systems Using GPUs

Desktop Computer

GPU and CPU each of their own memory.

Communicate over high-speed bus. (Often, PCI Express)

CPU in charge.



GPU v. CPU

GPU can execute floating-point operations faster.

GPU can read and write memory faster.

CPUs are easier to program.

CPUs can run certain programs faster.

Why GPUs are important.

They can do certain things much better than a CPU.

They have succeeded where many have failed.

May be only viable way of overcoming fundamental barriers to faster computation.

GPUs succeeded where many failed: Establishing a new computer architecture.

CPUs today share similar architecture.

Are designed for multiple uses: business, scientific.

Exceptions are minor: embedded.

In the past specialized architectures could be successful.

Cray supercomputers.

In most cases, different or specialized architectures failed:

Database machines.

LISP machines.

Alternatives failed because:

They were not that much faster.

Were very expensive to develop.

Market was small and so engineering big part of cost.

Reason GPUs succeeded:

First, their birth.

3D computer graphics is compute intensive. (Think frame rate.)

Need.

Graphics computation is well structured.

Facilitates development of specialized processor.

Amount of code relatively small.

Can be isolated in libraries.

Large market: Home computers, game consoles.

Can amortize development costs.

Their evolution to non-graphical use.

Computational characteristics of 3D graphics code shared by other types of programs.

GPUs could easily be modified to support non-graphical use.

CPU /GPU Similarities

Both run programs.

Machine languages are similar.

Both have instructions for integer and FP operations.

High-end chips are roughly same area, and draw same power.

GPU Incidental (and diminishing) Differences

Special hardware for *texture fetch and filtering* operations.

Special hardware for *interpolation*.

Trend is less specialized hardware.

GPU Important Performance Spec Differences

GPUs can do more FP operations per second.

GPUs can transfer more data per second.

GPU Programmability Differences

Extensive tuning required to achieve performance.

GPUs perform poorly on certain problems, regardless of tuning.