GPGPU
Introduction

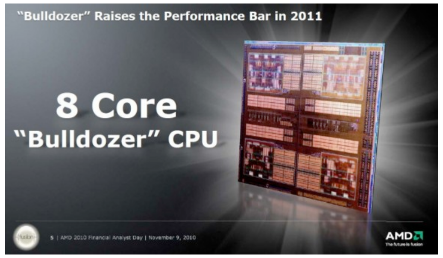
Alan Gray

EPCC

The University of Edinburgh

Introduction

- Central Processing Unit (CPU) of a computer system must be able to perform a wide variety of tasks efficiently.
- Until (relatively) recently, most CPUs comprised of 1 sophisticated compute core (for arithmetic), plus complex arrangement of controllers, memory caches, etc
- Increases in CPU performance were achieved through increases in the clock frequency of the core.
  - This has now reached it’s limit mainly due to power requirements
- Today, processor cores are not getting any faster, but instead we are getting increasing numbers of cores per chip.
  - Plus other forms of parallelism such as SSE,AVX vector instruction support



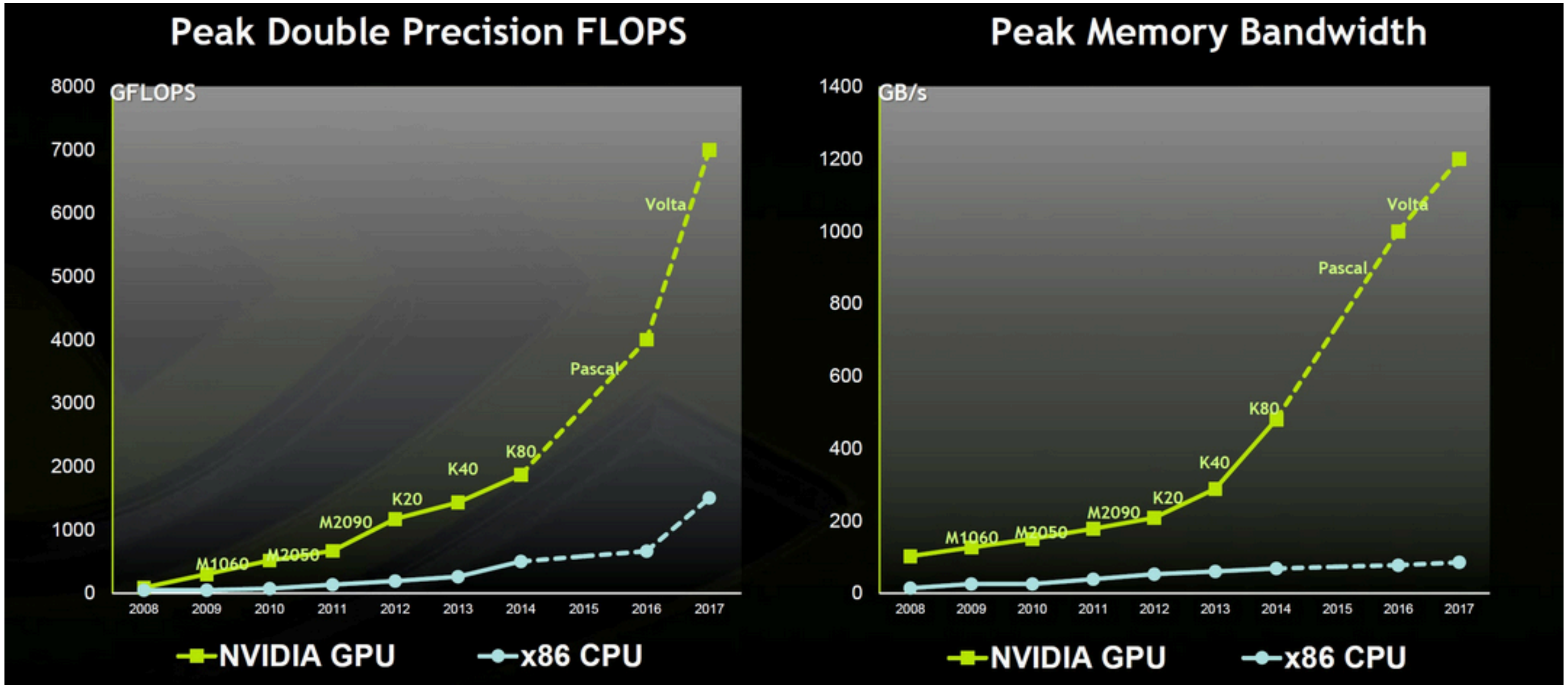
- Harder for applications to exploit such technology advances.

Introduction

Meanwhile….

- In recent years computer gaming industry has driven development of a different type of chip: the Graphics Processing Unit (GPU)
- Silicon largely dedicated to high numbers (hundreds) of simplistic cores,
  - at the expense of controllers, caches, sophistication etc
- GPUs work in tandem with the CPU (communicating over PCIe), and are responsible for generating the graphical output display
  - Computing pixel values
- Inherently parallel - each core computes a certain set of pixels
- Architecture has evolved for this purpose

Introduction

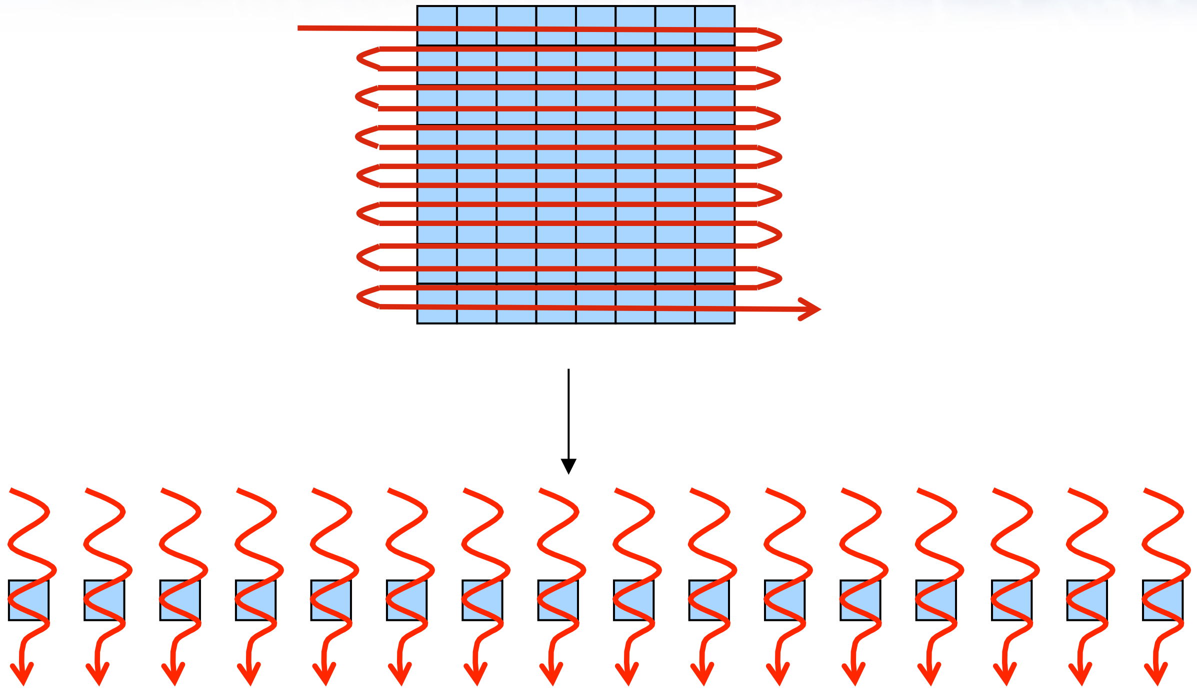


- GPU performance has been increasing much more rapidly than CPU
- Can we use GPUs for general purpose computation?
  - Yes (with some effort).

GPGPU

- GPGPU: General Purpose computation on Graphics Processing Units.
- GPU acts as an “accelerator” to the CPU (heterogeneous system)
  - Most lines of code are executed on the CPU (serial computing)
  - Key computational kernels are executed on the GPU (stream computing)
    - Taking advantage of the large number of cores and high graphics memory bandwidth
  - AIM: code performs better than use of CPU alone.
- GPUs now firmly established in HPC industry
  - Can augment each node of parallel system with GPUs

GPGPU: Stream Computing



- Data set decomposed into a *stream* of elements
- A single computational function (*kernel*) operates on each element
  - "thread" defined as execution of kernel on one data element
- Multiple cores can process multiple elements in parallel
  - i.e. many threads running in parallel
- Suitable for data-parallel problems

Programming Considerations

- Standard (CPU) code will not run on a GPU unless it is adapted
- Programmer must
  - decompose problem onto the hardware in a specific way (e.g. using a hierarchical thread/grid model in CUDA)
  - Manage data transfers between the separate CPU and GPU memory spaces.
  - Traditional language (C, C++, Fortran etc) not enough, need extensions, directives, or new language.
- Once code is ported to GPU, optimization work is usually required to tailor it to the hardware and achieve good performance
- Many researchers are now successfully exploiting GPUs
  - Across a wide range of application areas