Project CrayOn: Back to the future for a more General-Purpose GPU?

Philip Machanick

RHODES UNIVERSITY
Grahamstown • 6140 • South Africa

Why a better GPU?

- makes a lot of sense to base HPC on mass-market parts
  - economy of scale, mass base for toolchains
- but: GPUs purpose-designed for graphics can be a poor fit to general workloads

what can we learn from the past?

GPU endpoint

- at some point a faster GPU will saturate human sense
  - after that making a GPU faster for GPGPU only makes sense on momentum
  - losing a little of ultimate GPU speed will matter less than being more generally applicable

CDC 6600

1965 – many ideas behind RISC

By Jitze Couperus - Flickr: Supercomputer - The Beginnings, CC BY 2.0, https://commons.wikimedia.org/w/index.php?curid=19382150

Cray-1

1975 – first successful vector machine

By Clemens PFEIFFER (Own work) [CC BY 2.5 (http://creativecommons.org/licenses/by/2.5) or CC BY 2.5 (http://creativecommons.org/licenses/by/2.5)], via Wikimedia Commons

a lot to learn from Seymour Cray and competition

What makes a supercomputer?

- more expensive mix of standard parts
  - many CPUs, faster memory and interconnects
- exploiting packaging breakthroughs
  - Cray-1 used SRAM
- novel architectures that suit HPC
  - e.g. vectors and other single instruction-multiple data stream (SIMD) modes

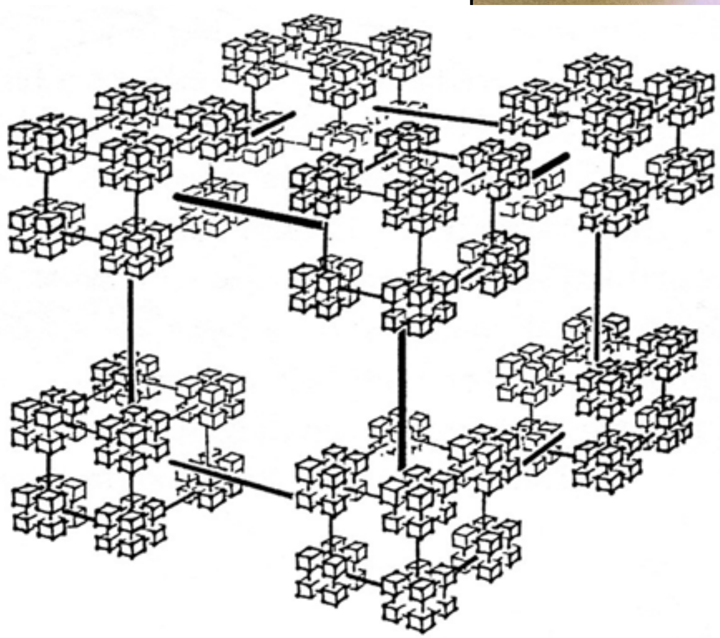
Weird and wonderful

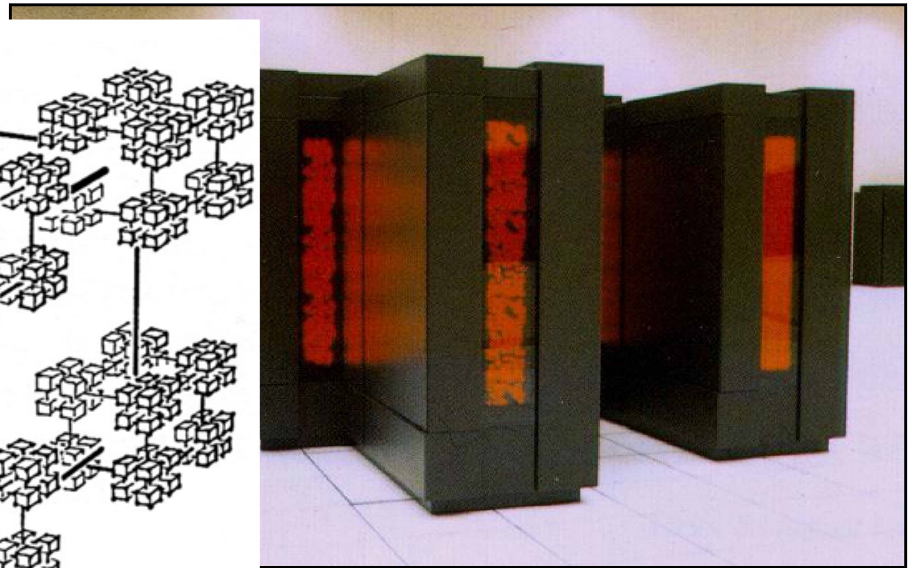
CM-1

1986 – up to 64Ki 1-bit processors
12-dimensional hypercube interconnect

CM-5

1991 – up to 64Ki Sparc processors





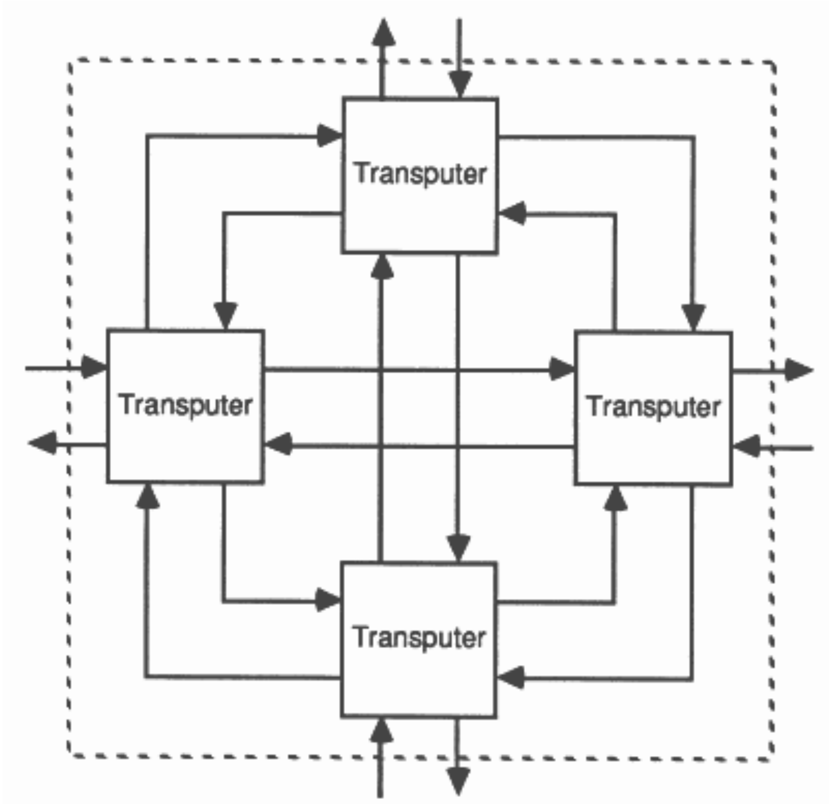
http://tamikothiel.com/cm/CM-1_r_700w.gif

http://tamikothiel.com/cm/press/TheDesignOfTheConnectionMachine.pdf

http://people.csail.mit.edu/bradley/cm5/Muzio_CM5.jpg

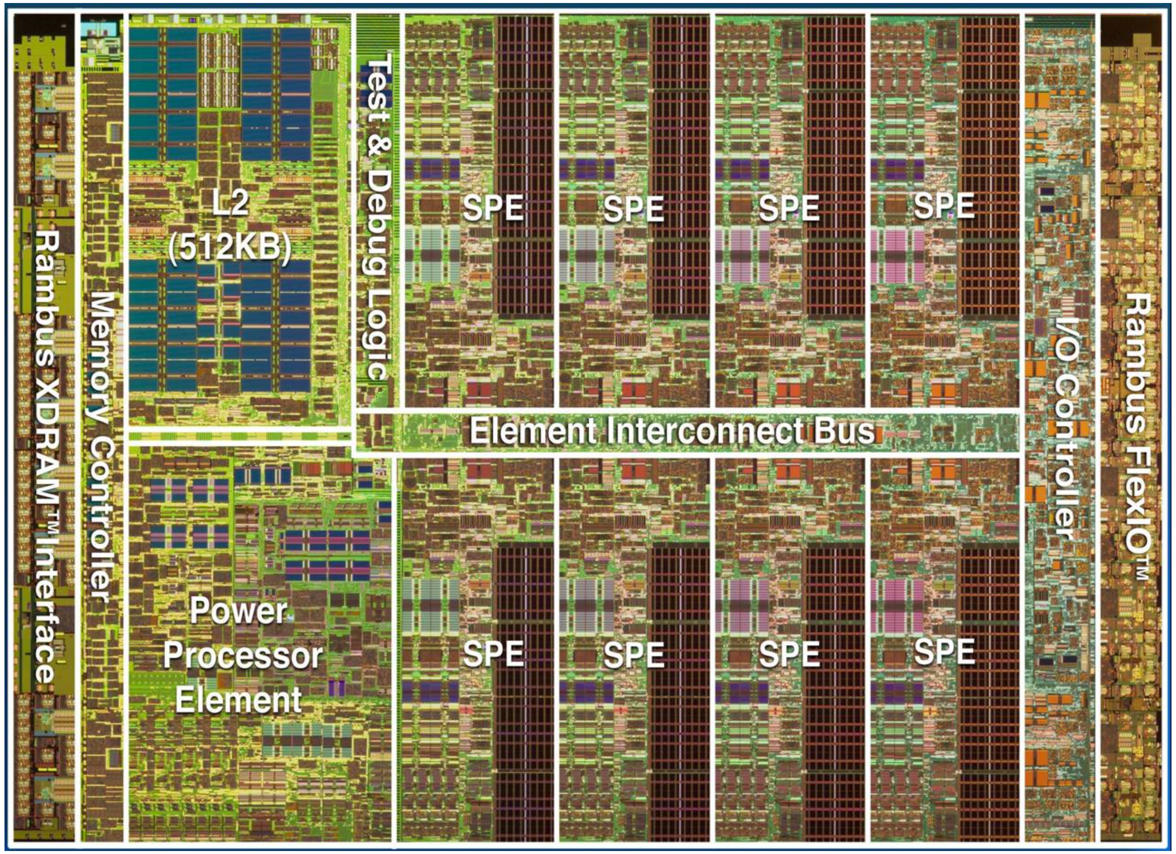
the ultimate SIMD design

Other ideas



http://www.transputer.net/fbooks/tarch/tarch.html

transputer – 4x high speed bidirectional serial links, distributed memory – 1980s



http://www.psdevwiki.com/ps3/CELL_BE

Cell Broadband Engine – e.g. used in PS3
1 PPC CPU, 8x vector+local memory units

General shake-down

- hard to program, limited fit to application space
  - SIMD
  - distributed memory
  - specialist local memory

Back to GPUs

- Ahead of GPU endpoint
  - reverse the design logic:
    - start with a good general-purpose design and add enough to do graphics

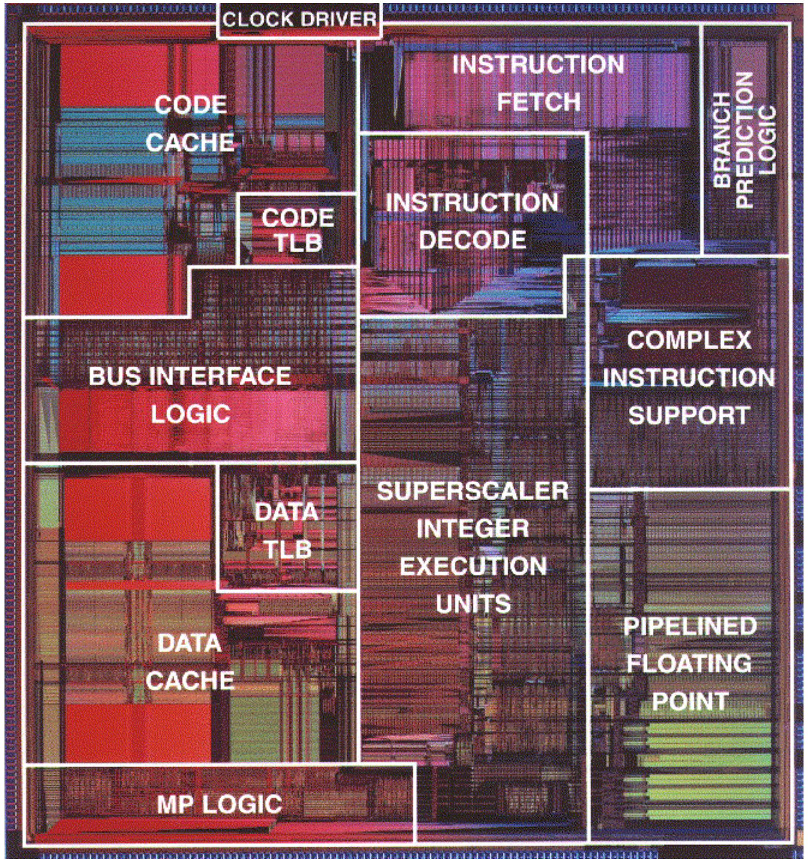
What about Larrabee?

- Intel project to do just that
    - started from Pentium pipeline
    - multiple cores
    - graphics extensions
- did not get to market

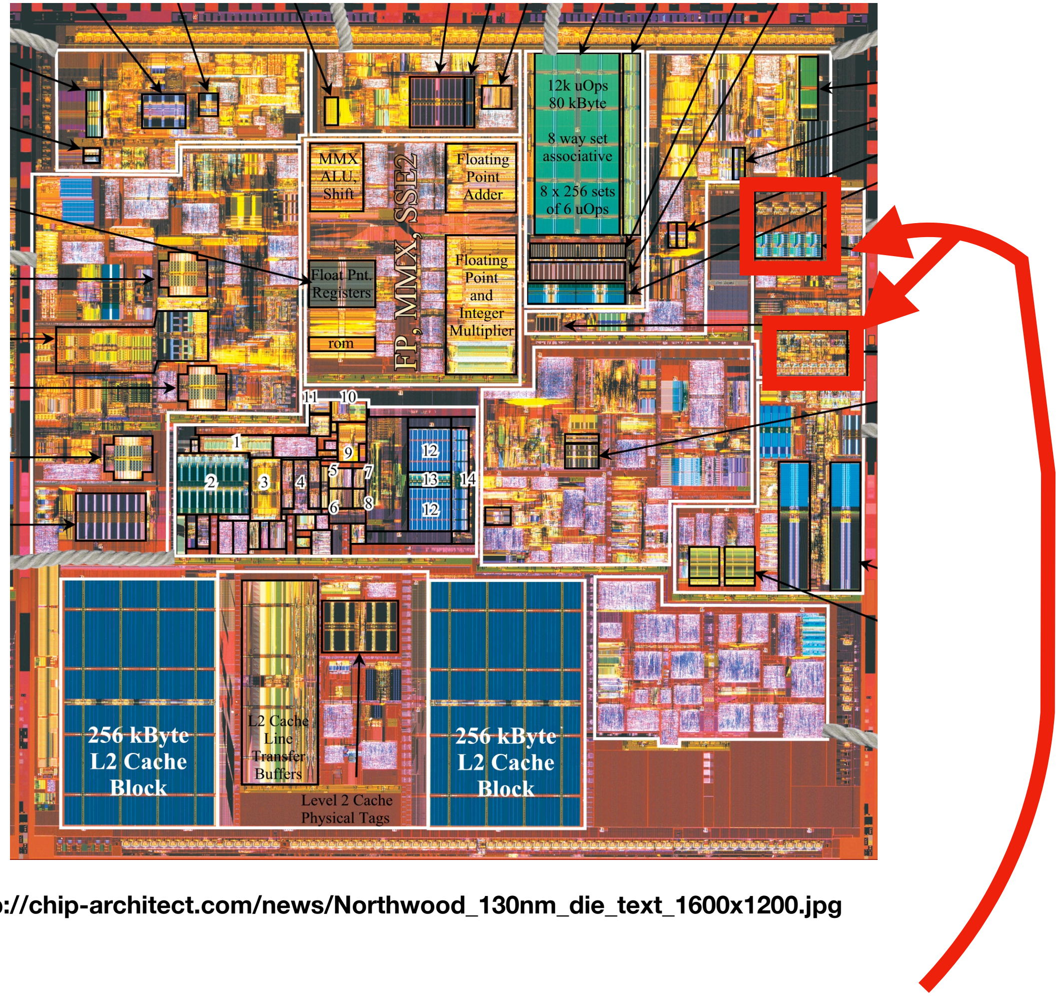
what went wrong?

Why Intel caught up

Pentium 1993



Pentium 4 Northwood 2002



https://people.cs.clemson.edu/~mark/330/colwell/pentium.gif

http://chip-architect.com/news/Northwood_130nm_die_text_1600x1200.jpg

Alpert, D., & Avnon, D. (1993). Architecture of the Pentium microprocessor. *IEEE micro*, 13(3), 11-21.

μ-ops created in decode – from there on close to RISC

Basic Pentium pipeline not so competitive

- to crack instructions into *μ*-ops is a small extra overhead when the backend is big and complex
    - *μ*-ops avoid directly executing CISC instructions in hardware
- a significant overhead with a simple pipeline

Bottom line?

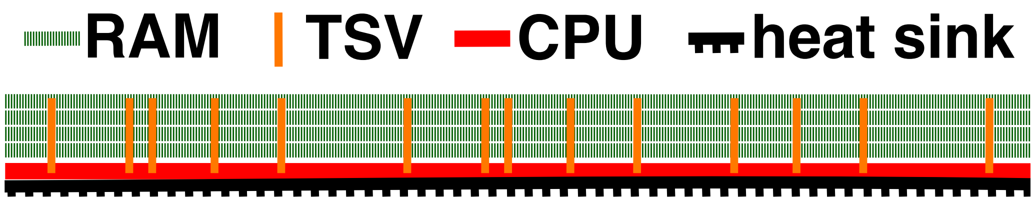
- If you want to build a multicore design with a large number of simple CPUs, RISC is still likely to win
- shared memory is much easier to program than distributed memory or specialist local memory
- multiple similar CPUs and vectors are the easiest modes of parallelism to exploit

Packaging

- Cray-1 was able to take advantage of a memory breakthrough
    - cylindrical shape reduced propagation delays
- what’s new?

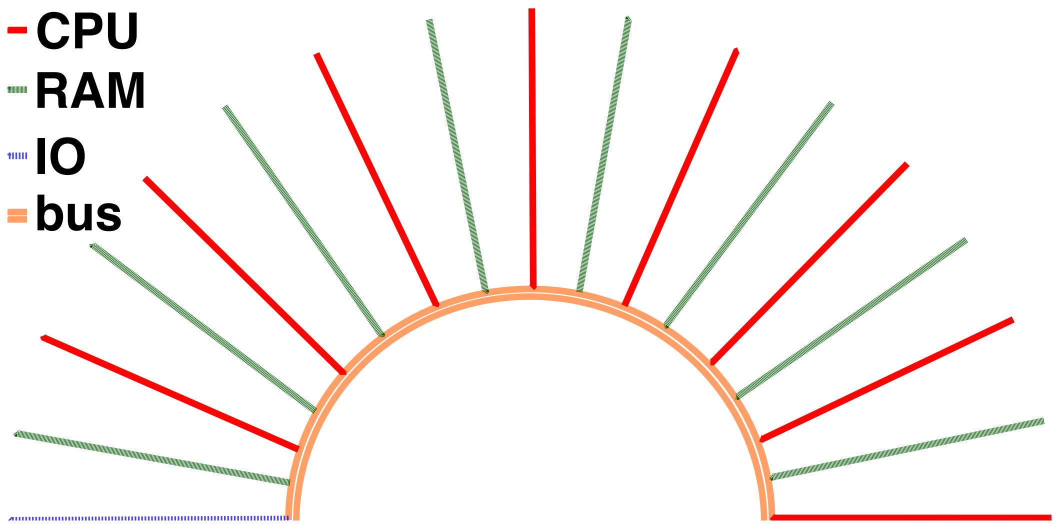
Going 3D

- Moore’s Law scalability limited by factors like leakage current
- 2.5D
  - 3D die stacking
    - picoserver
    - HMC RAM – logic layer plus DRAM



Even more 3D

- 3D Xpoint NVRAM has 3D internal structure
- what of recreating the Cray-1 structure in miniature?



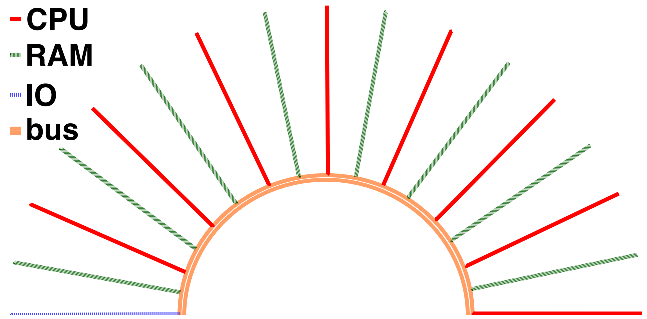
propagation delays short, can be made relatively precise – ideally avoid buffering

space between dies and under interconnect for cooling

Putting it all together

- GPGPU is a good idea – mass-market base for HPC part
- start from GP then add GPU: GP+GPU
- what will a GP+GPU look like?

CrayOn



Each CPU module: multiple simple RISCs with vector units

RAM modules: either SRAM or fast DRAM

SRAM configurable as caches or specialist graphics RAM

what else does a GPU need?