NYU
New York University

CSCI-GA.3033-012
Multicore Processors: Architecture & Programming

Lecture 10: Heterogeneous Multicore

Mohamed Zahran (aka Z)
mzahran@cs.nyu.edu
http://www.mzahran.com

Status Quo

- Previously, CPU vendors tried to use special purpose units for targeted performance wins:
  - FPUs, SSEs, Altivec, …
- Now industry explores hybrid models
  - Cell: 1 PowerPC and 8 SPEs
  - AMD Fusion: CPU + GPU
  - Intel Sandy Bridge
  - …

Classification of Heterogeneous Multicore

	Same cores	Different cores
Different ISAs	?	• Some Systems on Chip • multi/manycore & GPUs
Same ISA	Traditional Multi/Many cores Homogeneous Multicore	Cores of Different Capabilities

How Do We Design Them?

- by designing a series of cores from scratch
- by reusing a series of previously-implemented processor cores and change the interface
  - The entire family of a microprocessor can be incorporated into a single chip (how about clock frequency?)
- a combination of the two approaches

Why do we need them?

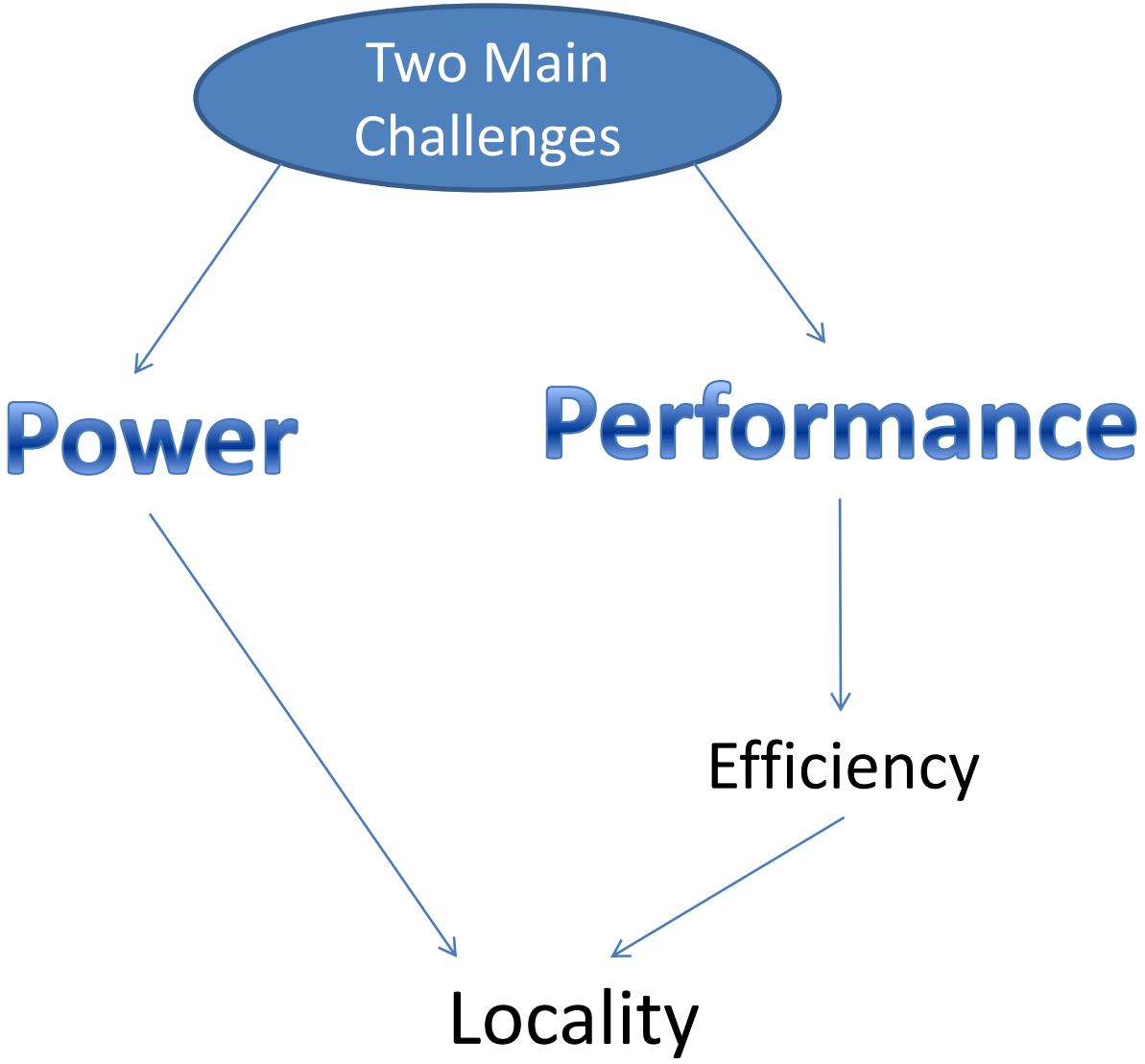
- Today's microprocessor design should balance high throughput and good single-thread performance
  - Haven't SMT and CMP achieved this?
- Heterogeneous multicore can do it more efficiently
  - Match each application to the core(s) of its performance demand
  - Provide better area-efficient coverage of workload demands

Why do we need them?

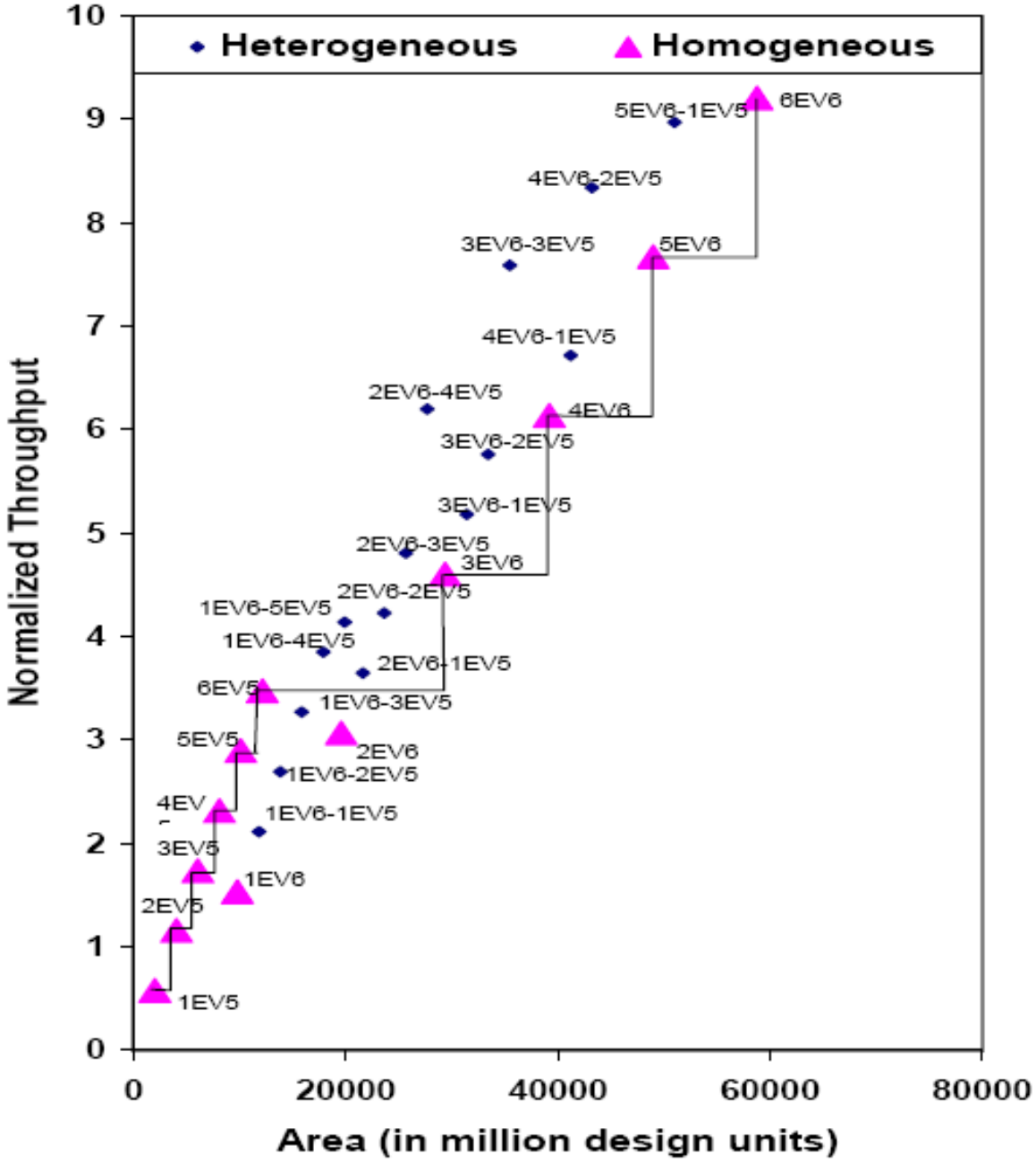
- Different applications have different resource requirements during their execution
  - What are these resource requirements?
  - The best core for execution may vary over time
- What do we gain from assigning the right thread/process/application to the right core?
  - performance enhancement
  - power consumption reduction

Why do we need them?

- Application with fewer and sophisticated threads -> Traditional multicore with latency optimized cores
- Application with high concurrency -> large number of throughput optimized cores



Data movement costs more than computation.

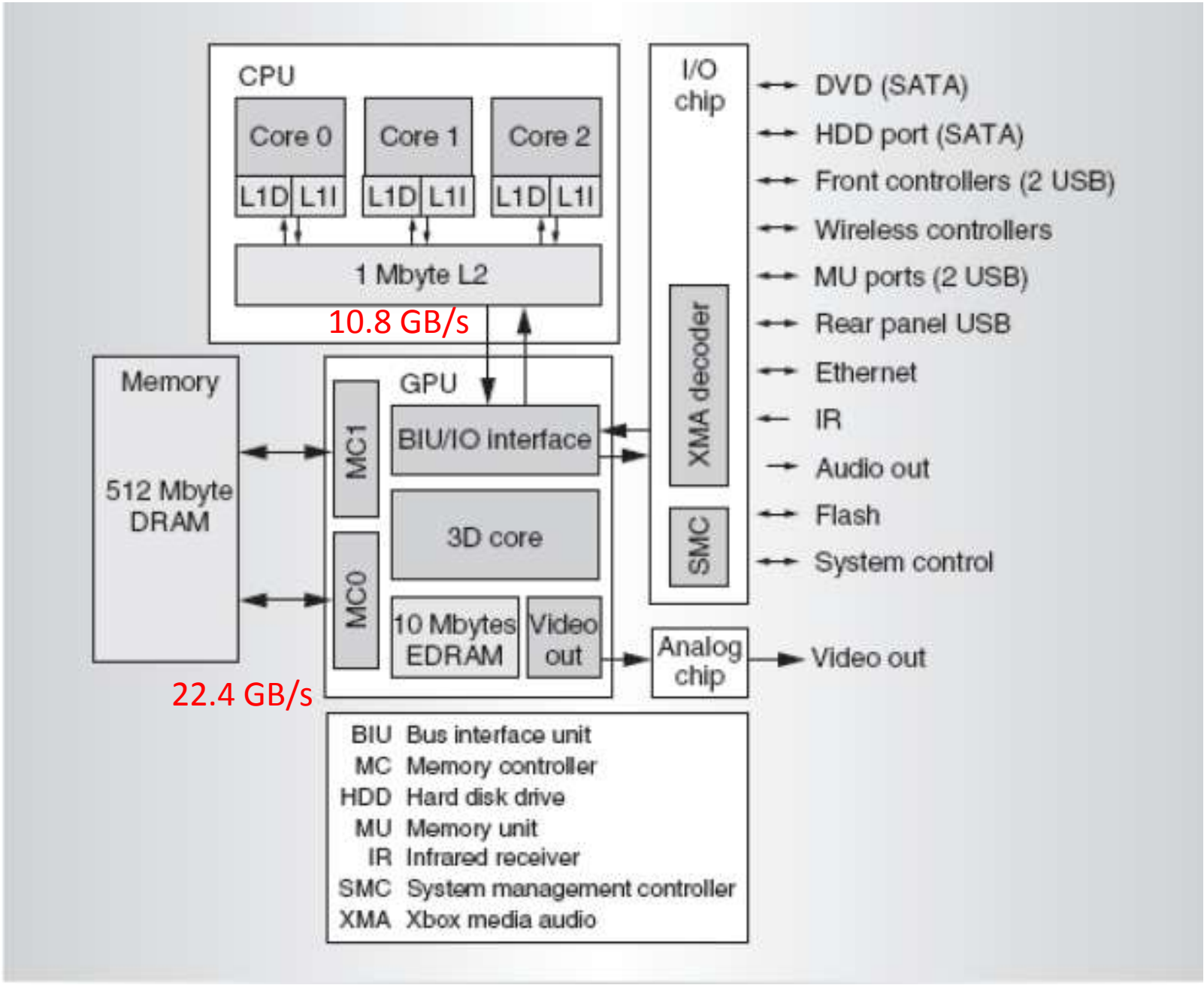

Heterogeneous
Homogeneous
Normalized Throughput
Area (in million design units)
0
1
2
3
4
5
6
7
8
9
10
0
20000
40000
60000
80000
6EV6
5EV6-1EV5
4EV6-2EV5
3EV6-3EV5
5EV6
4EV6-1EV5
2EV6-4EV5
4EV6
3EV6-2EV5
3EV6-1EV5
2EV6-3EV5
3EV6
2EV6-2EV5
1EV6-5EV5
1EV6-4EV5
2EV6-1EV5
6EV5
1EV6-3EV5
5EV5
2EV6
1EV6-2EV5
4EV
1EV6-1EV5
3EV5
1EV6
2EV5
1EV5

Issues in Heterogeneous Multicore Processors

- Are they scalable?
- How is the scheduling done (objective function)?
- Who does the scheduling?
- What is the cost of task switching?
- What is the cost of task migration?
- When to switch core?
- Still need shared address space?
- How about coherence?

Examples from real-life

XBOX 360


CPU
Core 0
Core 1
Core 2
L1D
L1I
1 Mbyte L2
10.8 GB/s
Memory
512 Mbyte DRAM
22.4 GB/s
GPU
MC1
MC0
BIU/IO interface
3D core
10 Mbytes EDRAM
Video out
I/O chip
XMA decoder
SMC
Analog chip
DVD (SATA)
HDD port (SATA)
Front controllers (2 USB)
Wireless controllers
MU ports (2 USB)
Rear panel USB
Ethernet
IR
Audio out
Flash
System control
Video out
BIU Bus interface unit
MC Memory controller
HDD Hard disk drive
MU Memory unit
IR Infrared receiver
SMC System management controller
XMA Xbox media audio

Specifications

- 3 CPU cores
  - 4-way SIMD vector units
  - 8-way 1MB L2 cache (3.2 GHz)
  - 2 way SMT
  - In-order
  - 2 Instructions/cycle
- ATI GPU with embedded EDRAM
- 3D graphics units
- 512-Mbyte DRAM main memory

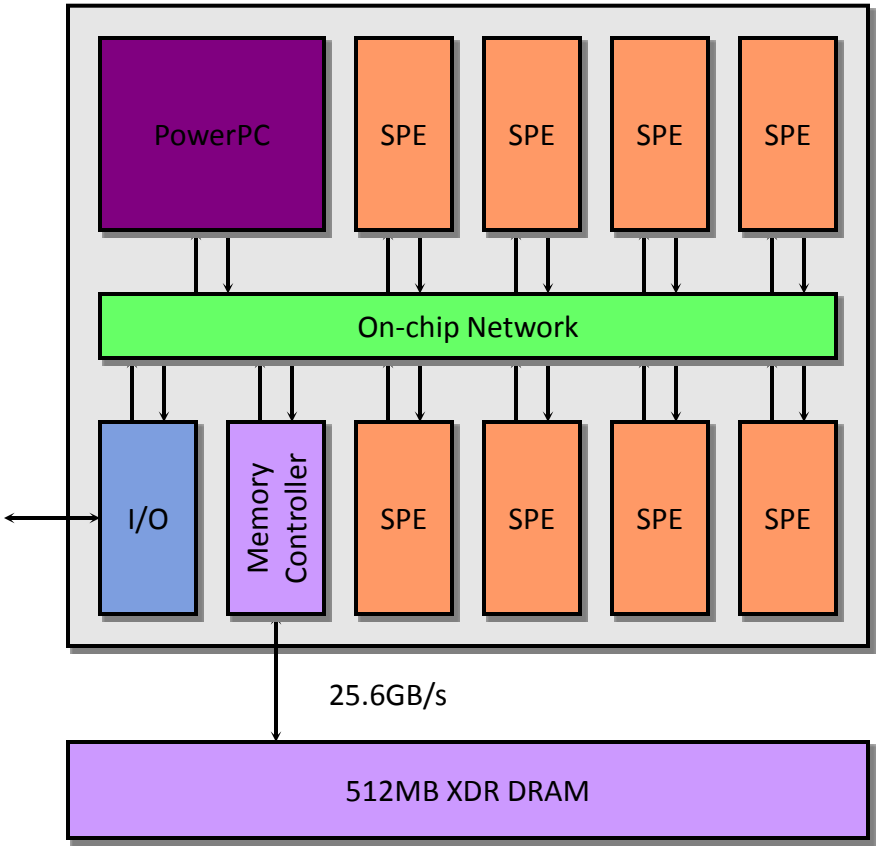
Philosophy

- Value for 5-7 years
- Big performance increase over last generation
- Support high-definition video
- extremely high pixel fill rate (goal: 100+ million pixels/s)
- Flexible to suit dynamic range of games
- balance hardware, homogenous resources
- Programmability (easy to program)

Cell Processor

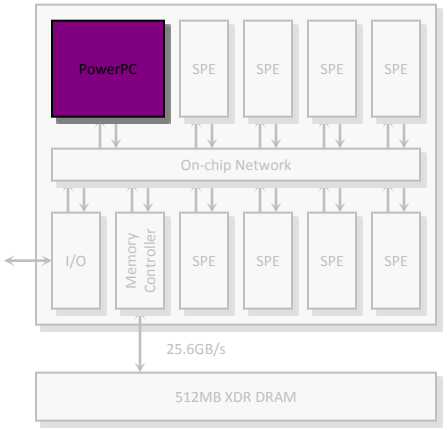
Overview

- Each Cell chip has:
  - One PowerPC core
  - 8 compute cores (SPEs)
  - On-chip Memory controller
  - On-chip I/O
  - On-chip network to connect them all
- A PS3 has
  - 1 Cell Chip (6 usable SPEs)
  - 256MB of DRAM Memory



PowerPC Core (PPE)

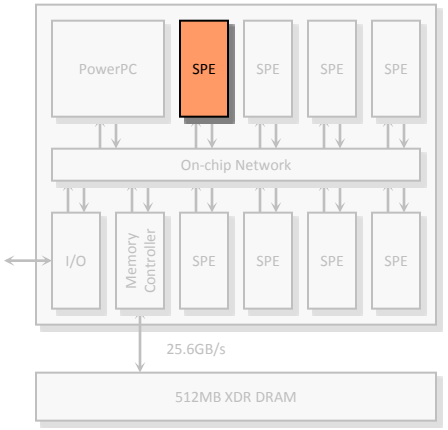
- 3.2GHz
- Dual issue, in-order
- 2-way multithreaded
- 512KB L2 cache
- No hardware prefetching
- SIMD (altivec) + FMA
- 6.4 GFlop/s (double precision)
- 25.6 GFlop/s (single precision)



- Serves 2 purposes:
  - Compatibility processor;
    - runs legacy code, compilers, libraries, etc.
  - Performs all system level functions
    - including starting the other cores

Synergistic Processing Element (SPE)

- Offload processor
- Dual issue, in-order, VLIW inspired SIMD processor
- 128 x 128b register file
- 256KB local store, no I$, no D$
- Runs a small program (placed in LS)
- Offloads system functions to the PPE
- MFC (memory flow controller)
  - a programmable DMA engine and on-chip network interface
- Performance
  - 1.83 GFlop/s (double precision)
  - 25.6 GFlop/s (single precision)
  - 51.2GB/s access the local store
  - 25.6GB/s access to the network



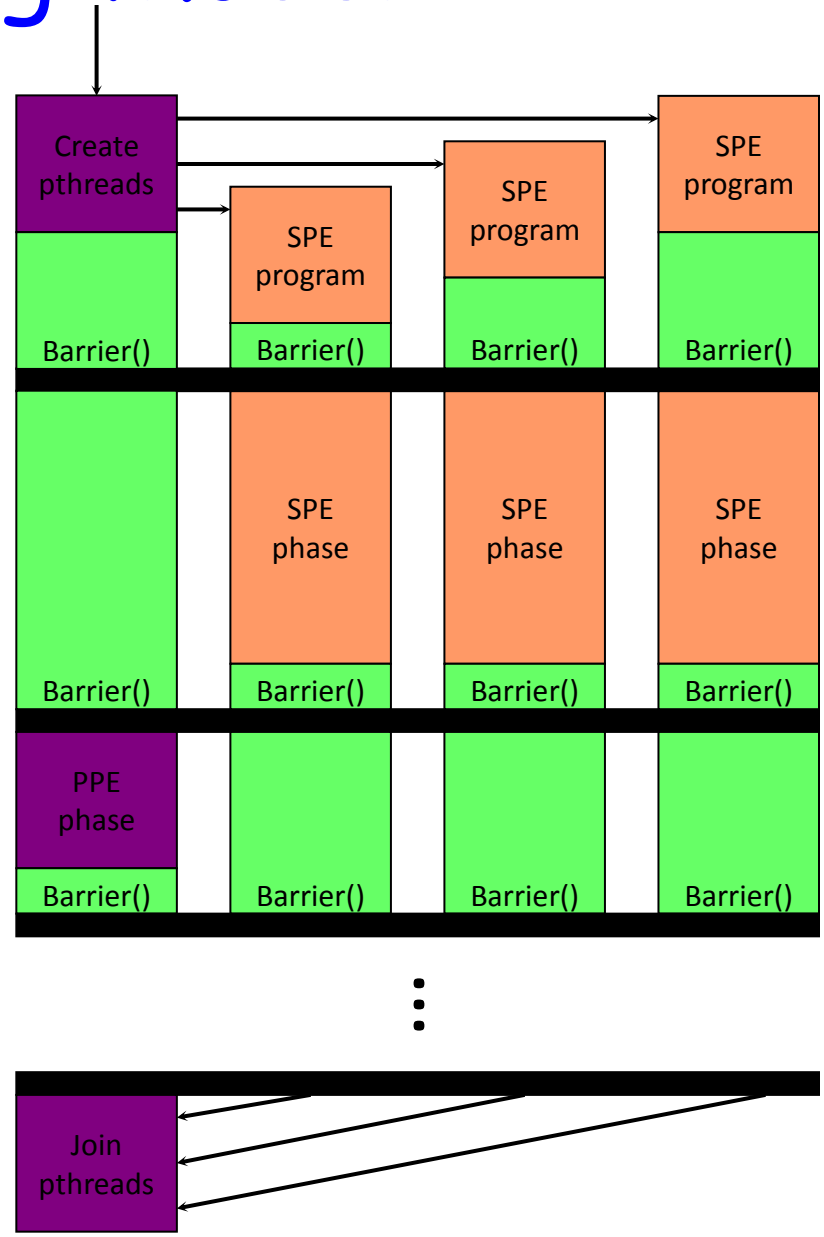
Threading-Model

- For each SPE program the PPE must:
  - Create a SPE context
  - Load the SPE program (embedded in the binary)
  - Create a pthread to run it
- Each pthread can be as simple as:

```
spe_context_run(…);
pthread_exit(…);
```

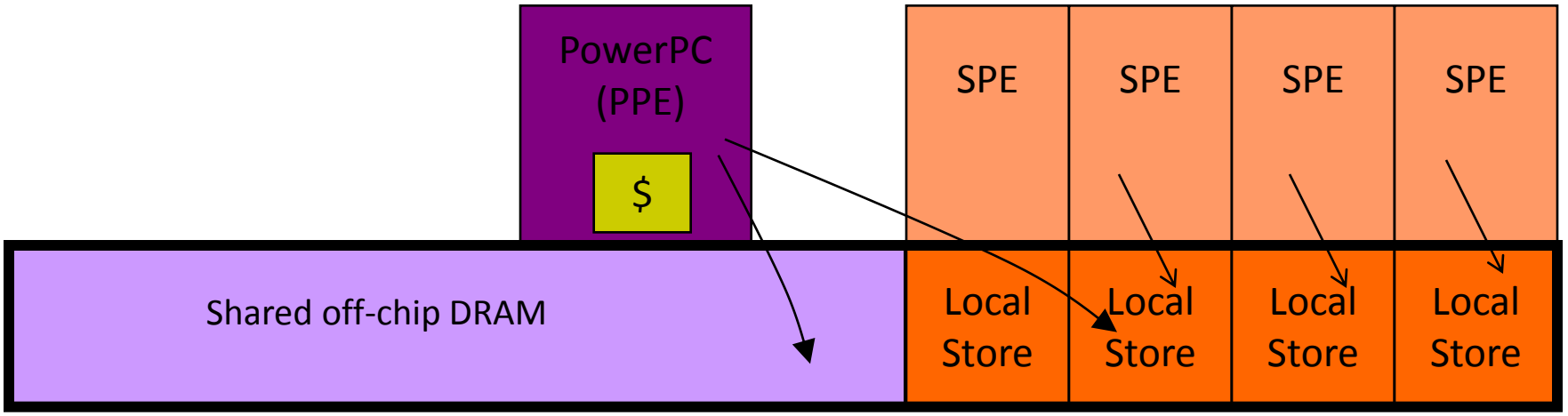
- Typically, split the work into 2 phases:
  - Code that runs on the SPEs
  - Code that runs on the PPE

with a barrier in between

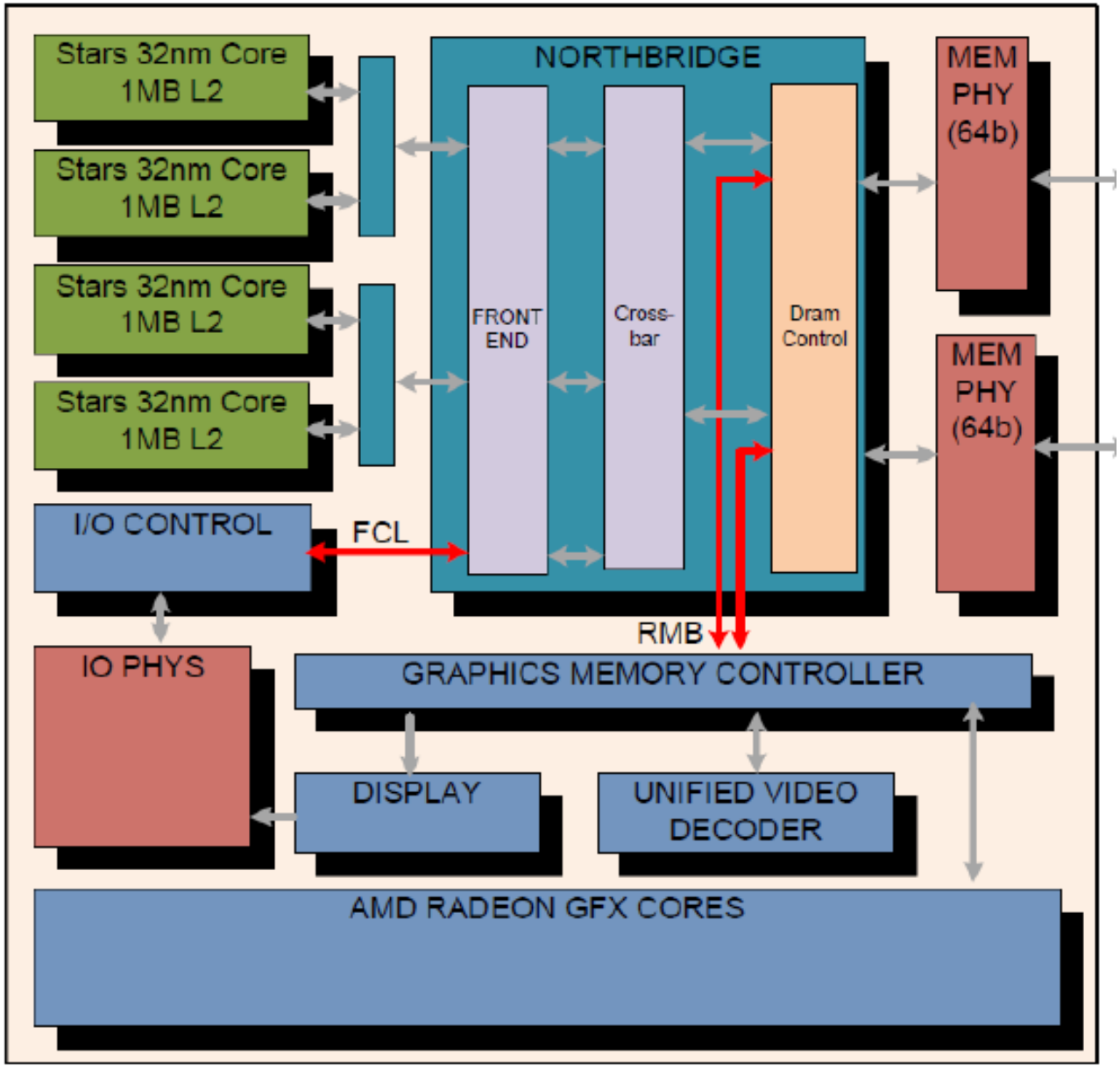


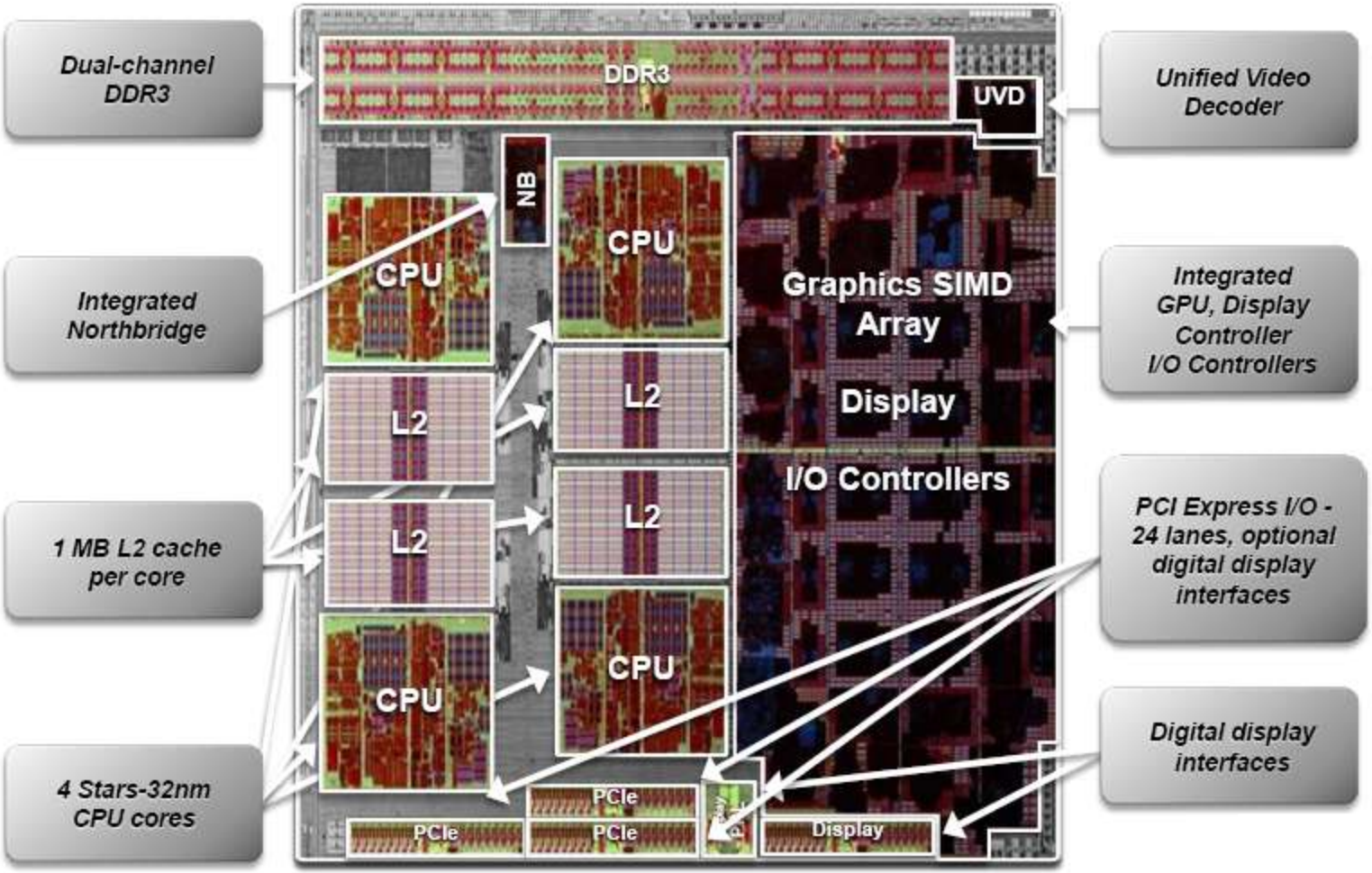
Threads-Communication

- The PPE may communicate with the SPEs via:
  - individual mailboxes (FIFO)
  - HW signals
  - DRAM
  - Direct PPE access to the SPEs' local stores



AMD'S "LLANO" FUSION APU

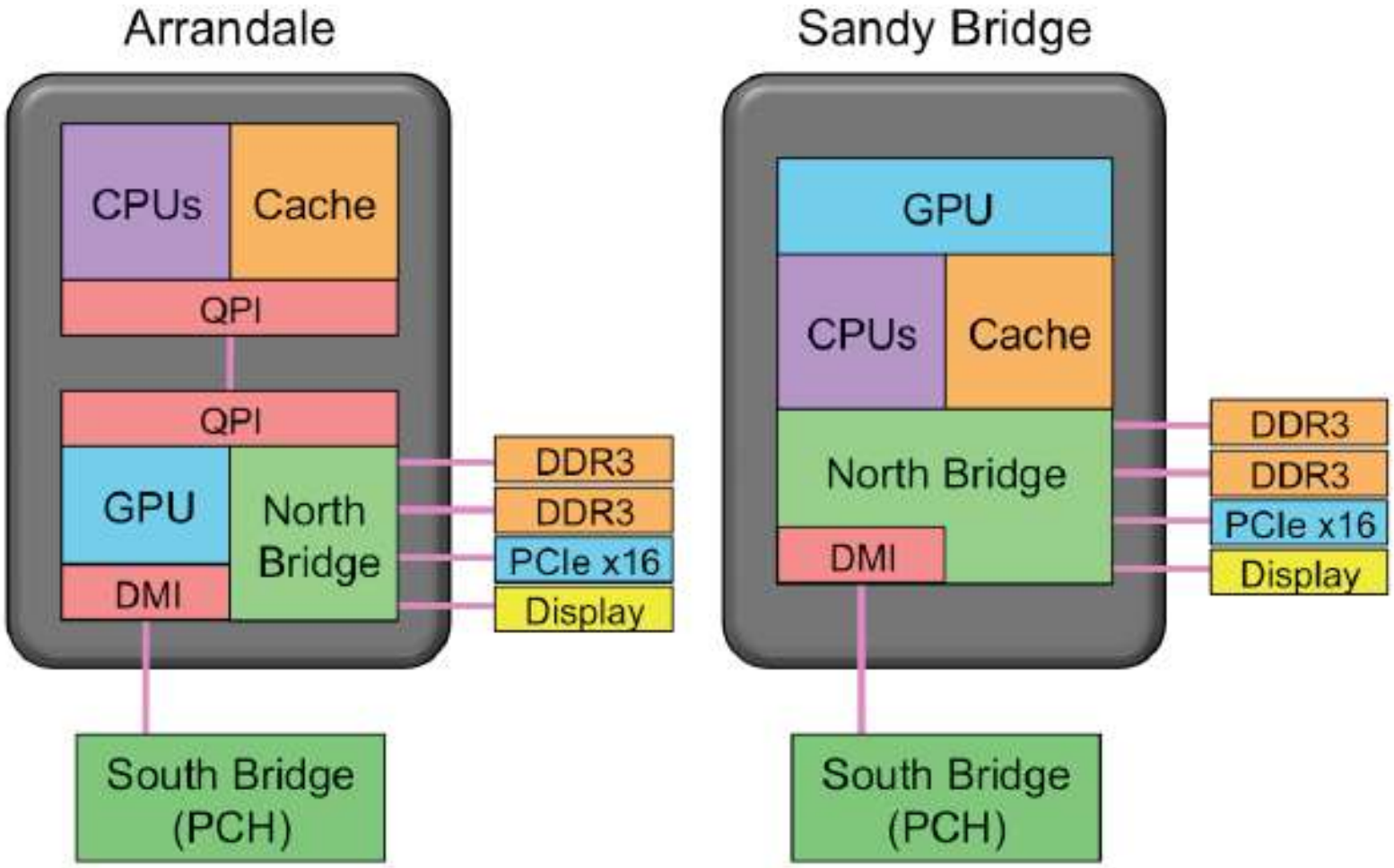

Stars 32nm Core
1MB L2
Stars 32nm Core
1MB L2
Stars 32nm Core
1MB L2
Stars 32nm Core
1MB L2
NORTHBRIDGE
FRONT END
Cross-bar
Dram Control
MEM PHY (64b)
MEM PHY (64b)
I/O CONTROL
FCL
RMB
IO PHYS
GRAPHICS MEMORY CONTROLLER
DISPLAY
UNIFIED VIDEO DECODER
AMD RADEON GFX CORES


Dual-channel DDR3
Integrated Northbridge
1 MB L2 cache per core
4 Stars-32nm CPU cores
DDR3
UVD
NB
CPU
CPU
CPU
CPU
L2
L2
L2
Graphics SIMD Array
Display
I/O Controllers
PCIe
PCIe
PCIe
Display
Unified Video Decoder
Integrated GPU, Display Controller I/O Controllers
PCI Express I/O - 24 lanes, optional digital display interfaces
Digital display interfaces

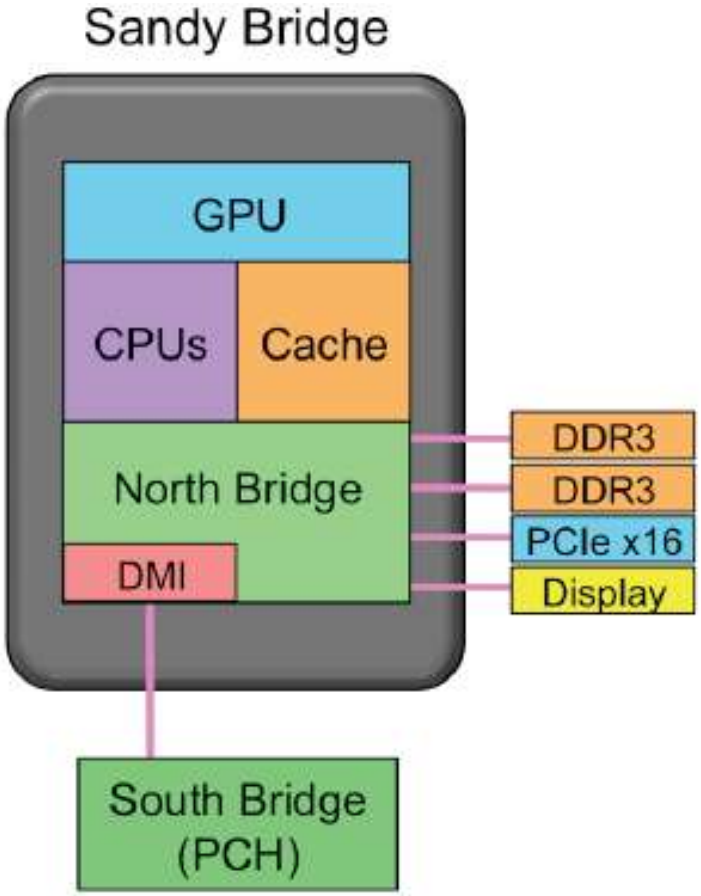
INTEL: SANDY BRIDGE

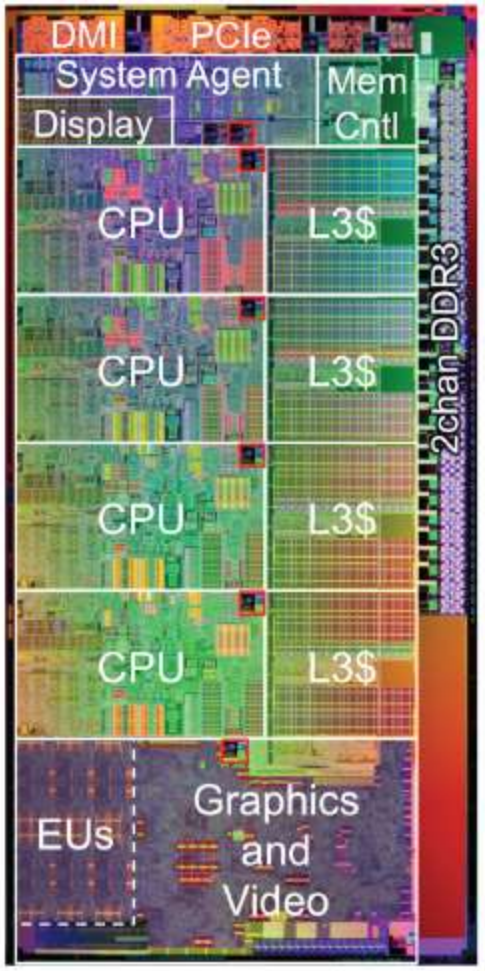
Sandy Bridge

- Intel Tock
  - Intel's first processor with GPU on the processor itself.
- Improvement over its predecessor Nehalem
- Targeting multimedia applications
  - Introduced Advanced Vector Extensions (AVX)
- More power-efficient than Westmere


Arrandale
CPUs
Cache
QPI
QPI
GPU
North Bridge
DMI
DDR3
DDR3
PCIe x16
Display
South Bridge (PCH)
Sandy Bridge
GPU
CPUs
Cache
North Bridge
DMI
DDR3
DDR3
PCIe x16
Display
South Bridge (PCH)

- The GPU can access the large L3 cache
- Intel's team totally re-designed the GPU





Example from research

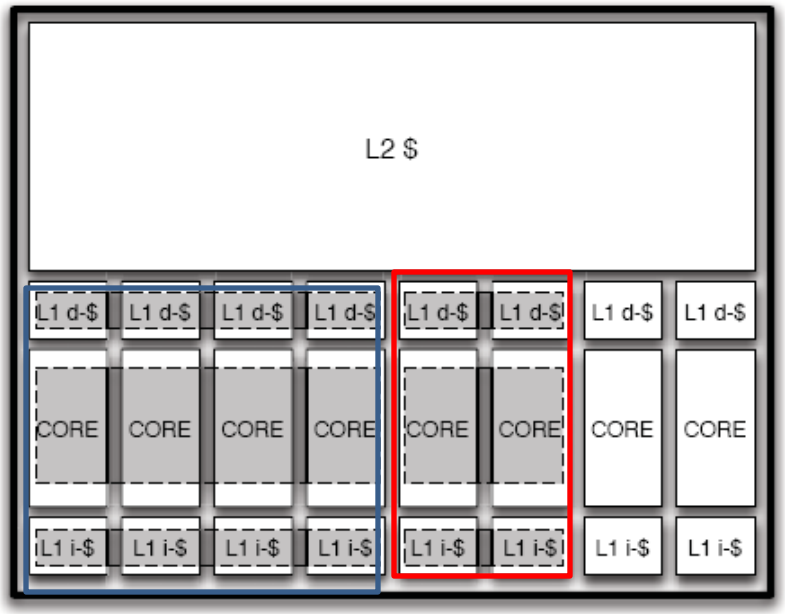
Core Fusion: Accommodating Software Diversity in Chip Multiprocessor

- A single application can go from highly sequential to highly parallel in different phases
- This is at odd with traditional multicore, or even tradition heterogeneous multicore
- Die composition is set at design time → and at design time we do not know the different applications that may execute on the chip!

What can we do?

Core Fusion: Accommodating Software Diversity in Chip Multiprocessor

- Group of independent cores
- Cores can dynamically morph into larger CPU or used as distinct CPUs.



Core Fusion: Accommodating Software Diversity in Chip Multiprocessor

- The application requests core fusion/split actions through a pair of *FUSE and SPLIT* ISA *instructions, respectively.*
- FUSE and SPLIT instructions are executed conditionally by hardware, based on the value of an OS-visible control register that indicates which cores within a fusion group are eligible for fusion.
- To enable core fusion, the OS allocates either two or four of the cores in a fusion group to the application when the applications context-switched in, and annotates the group's control register.

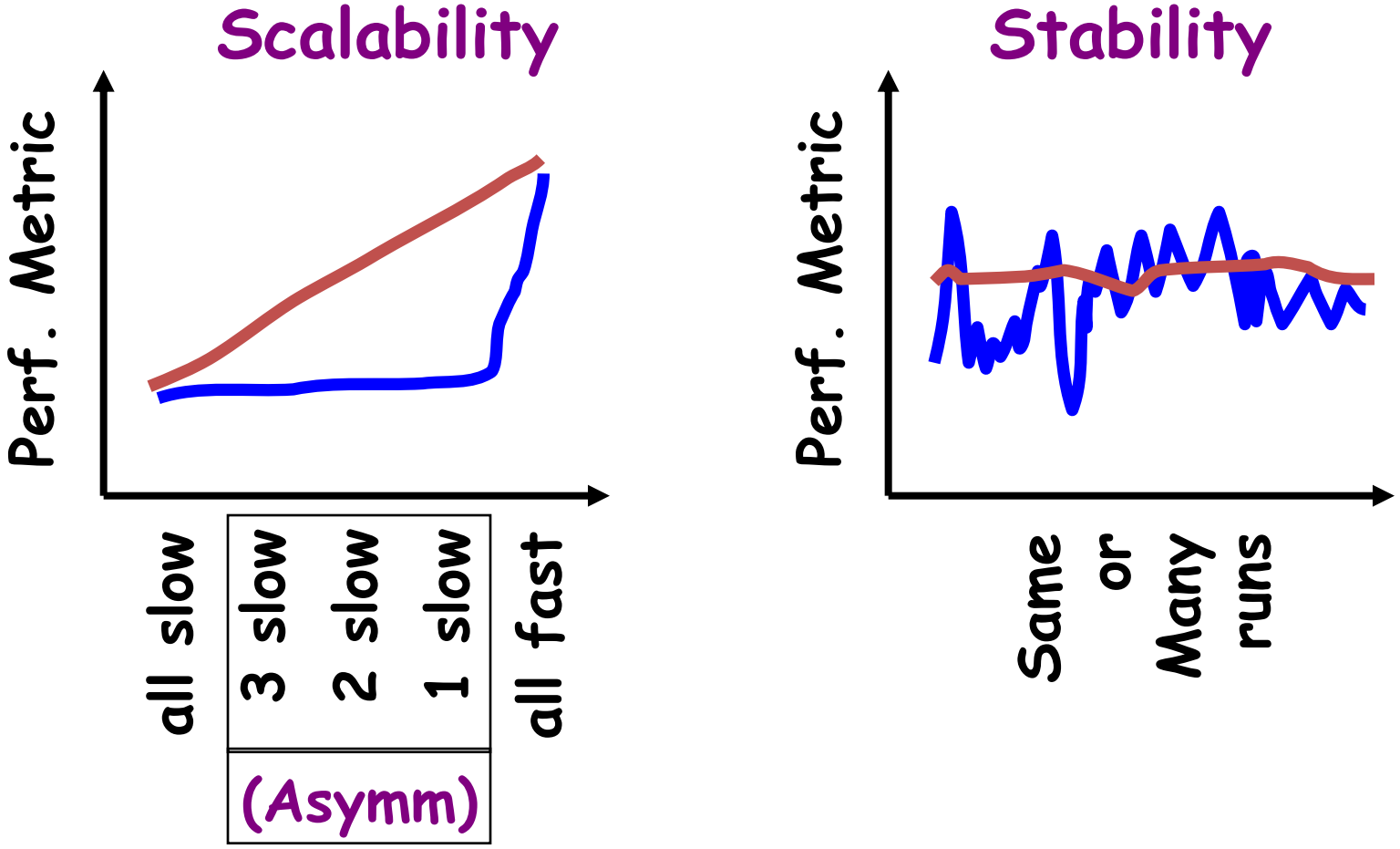
Challenges

- Programs demands vary over time
- Objective function may vary over time
- The core capability may vary over time (e.g. if it gets overheated)

Some Numbers

- In paper published in 2005: "The Impact of Performance Asymmetry in Multicore Architectures" … ISCA'05
- They experimented with 4 cores
- Used several configurations and many benchmarks
- What did they find?

Studying impact



Workloads evaluated

SPECjbb SPECjAppServer	Middle-tier business apps. Throughput parallel
Apache Zeus	Webservers Throughput parallel
TPC-H SPECOMP H.264	Task-based parallelization
PMake	Embarrassingly parallel

Impact of asymmetry

Workloads	Scalable	Stable	Fix
SPECjbb	✓	x	✓
SPECjAppServer	✓	✓	
Apache	✓	x	✓
Zeus	x	x	x
TPC-H	✓	x	✓
SPECOMP	x	x	✓
H.264	✓	✓	
PMake	✓	✓	

Programming Heterogeneous Multicore Systems

Programmers

Don't reason about asymmetry

Asymmetry negatively affects applications

-Observed unpredictable workload behavior

This can be fixed by

- Evaluating threads' **work partitioning**

-**Scheduling** of threads with asymmetry

As A Programmer

- Define your threads
  - As many threads as you can
- Know your hardware and your system software
  - How many cores?
  - How heterogeneous are they?
  - How does the OS scheduling work?
- Now combine your threads depending on the core strength
  - You can assign a thread to a core (many languages allow this: pthreads, OpenMP, ..)
  - Load imbalance is your enemy!

Load Imbalance in Parallel Applications: Main Reasons

- Poor single processor performance
  - Typically in the memory system
- Too much parallelism overhead
  - Thread creation, synchronization, communication
- Different amounts of work across processors
    - Computation and communication
- Different speeds (or available resources) for the processors

How To Recognize Load Imbalance?

- **Hint:** Time spent at synchronization is high and is uneven across cores, but not always so simple …
- Imbalance may change over phases!
- Insufficient parallelism always lead to load imbalance
- Useful tools:
  - http://www.cs.uoregon.edu/Research/tau/home.php
  - http://icl.cs.utk.edu/papi/
  - http://pages.cs.wisc.edu/~paradyn/

Important Questions When Dealing With Load Imbalance

- **Thread costs**
  - Do all threads have equal costs?
  - If not, when are the costs known?
    - Before starting, when created, or only when it ends
- **Thread dependencies**
  - Can all threads be run in any order (including parallel)?
  - If not, when are the dependencies known?
    - Before starting, when created, or only when it ends
- **Locality**
  - Is it important for some threads to be scheduled on the same processor (or nearby) to reduce communication cost?
  - When is the information about communication known?

Important Questions When Dealing With Load Imbalance

- Thread cost known before starting:
  - matrix multiplication
- Cost known when thread created:
  - N-Body
- Cost known when thread ends:
  - search

Important Questions When Dealing With Load Imbalance

- Threads can execute at any order:
  - dependence free loops
- Threads have a known dependency graph:
  - matrix computations
- Dependence unknown:
  - search

Important Questions When Dealing With Load Imbalance

- Threads, once created do not communicate:
  - embarrassingly parallel applications
- Threads communicate in predictable pattern:
  - PDE solver
- Information about communication is unpredictable:
  - discrete event simulation

Many solutions … Of Different Flavors

- Task queue (centralized or distributed)
- Work stealing
- Work pushing
- Off-line scheduling
- Self-scheduling
- …

Question: Are homogeneous cores of different ISAs useful in something? Justify

Conclusions

- Asymmetric systems may be the de facto in the future
- Good for energy and performance but may introduce unpredictability
- To reduce unpredictability, load-balancing is your friend!
- In multiprogramming environment, the advantage of heterogeneous multicore is obvious!