Multi-Core Architectures

15-213/18-243: Introduction to Computer Systems

27th (and last) Lecture, 28 April 2011

Instructors:

Anthony Rowe and Gregory Kesden

Today

- Final Exam
- Multi-core
- Parallelism on multi-core

Final Exam (Tuesday May 3rd, 1-4pm)

- Lecture 1: PH 100 & PH 125C
- Lecture 2: MI Mellon Auditorium

- Everything from lecture, the book and labs is fair game…
- Closed-Book, Close-Notes
- We will provide Exam 1 and Exam 2 Note Sheets

- **Exam Review Session: WEH 7500 Sunday 6-8pm**
  - **Email Questions to 15213review@gmail.com**

Final Topics (from Exam 1)

- Assembly Code
  - Assembly to C translation
  - Stack management
- Structure Alignment
- Ints, Floats, Boats
- Cache

Final Topics (from Exam 2)

- Process Control
- Signals
- File I/O
- Memory Layout
- Dynamic Memory

Final Topics (New)

- Concurrency + Thread Safety
  - Reader / Writer Locks, mutex, semaphore
  - Starvation, deadlock
  - What makes a function thread-safe vs unsafe?
- Networking
  - Understand notion of sockets and ports

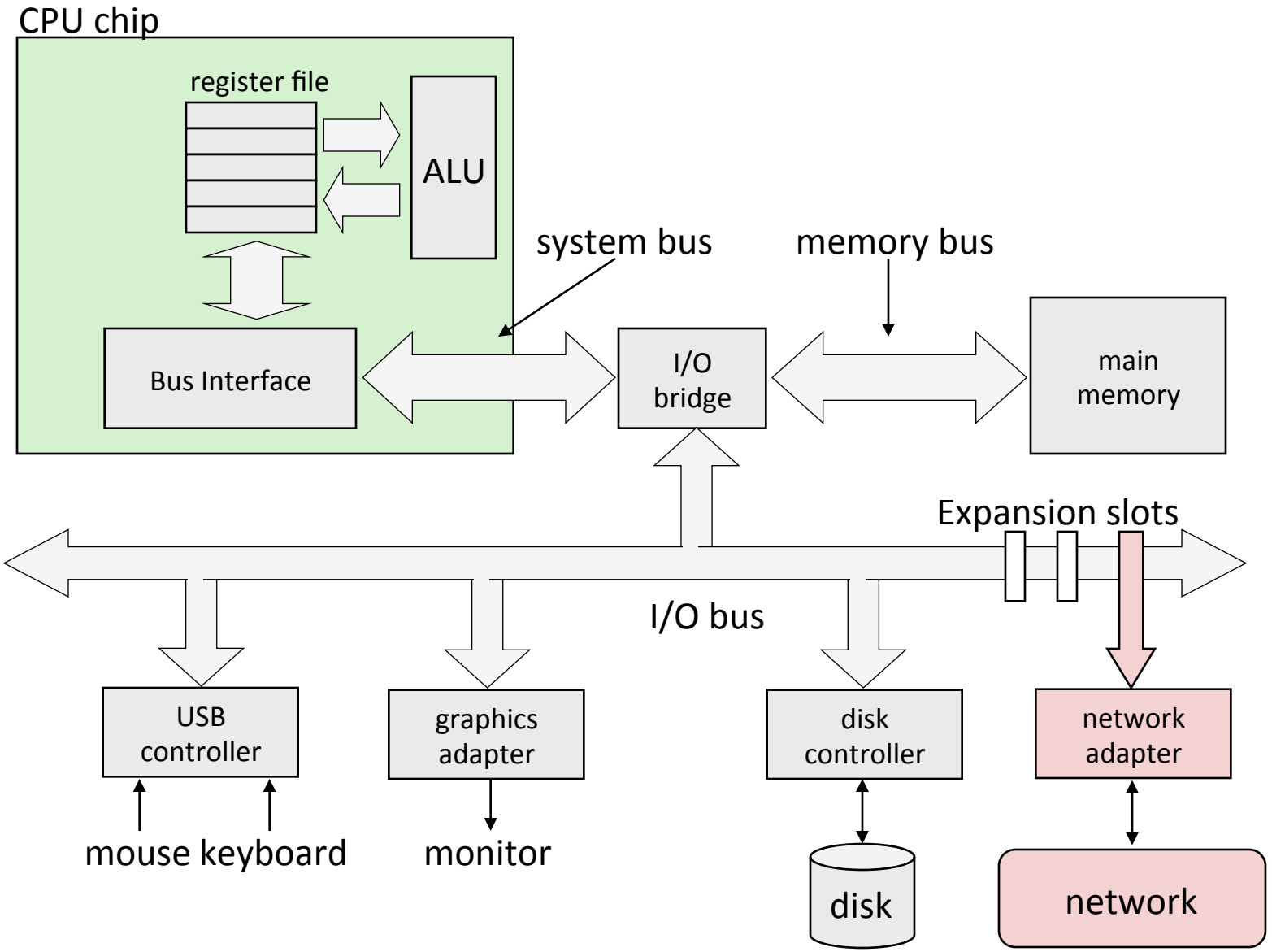
Today

- Multi-core
- Parallelism on multi-core

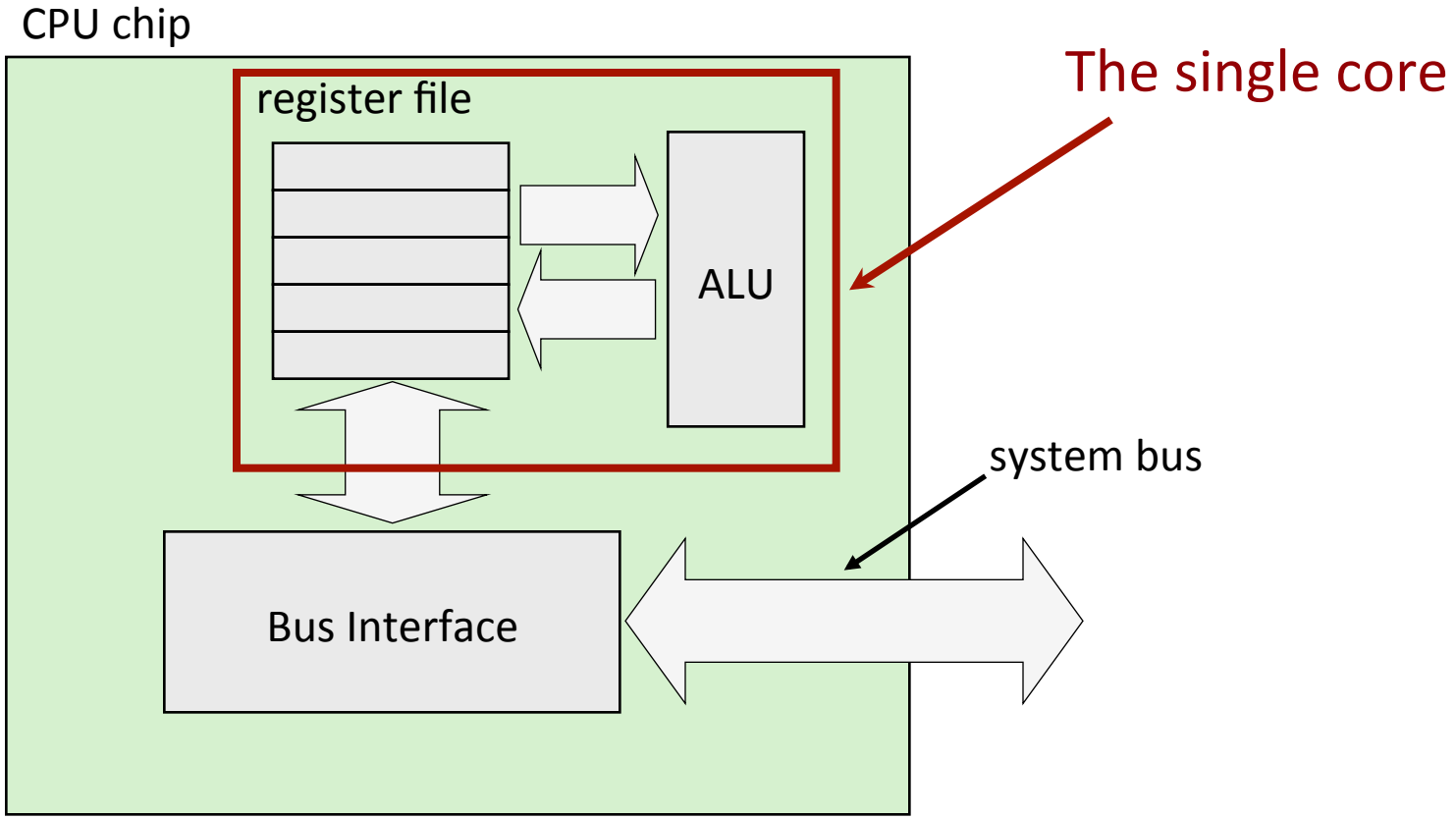
Why Multi-Core?

- Traditionally, single core performance is improved by increasing the clock frequency...
- ...and making deeply pipelined circuits...
- Which leads to...
  - Heat problems
  - Speed of light problems
  - Difficult design and verification
  - Large design teams
  - Big fans, heat sinks
  - Expensive air-conditioning on server farms
- Increasing clock frequency no longer the way to go forward

Single Core Computer

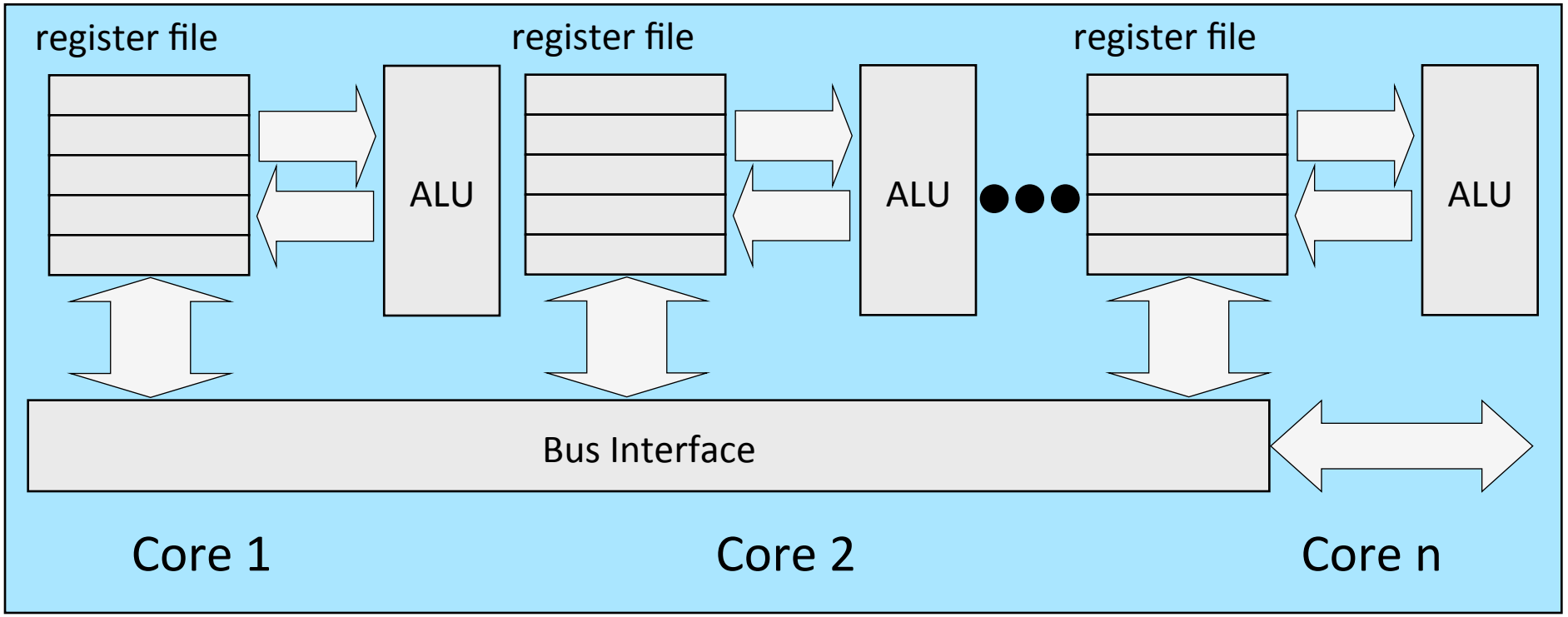


Single Core CPU Chip



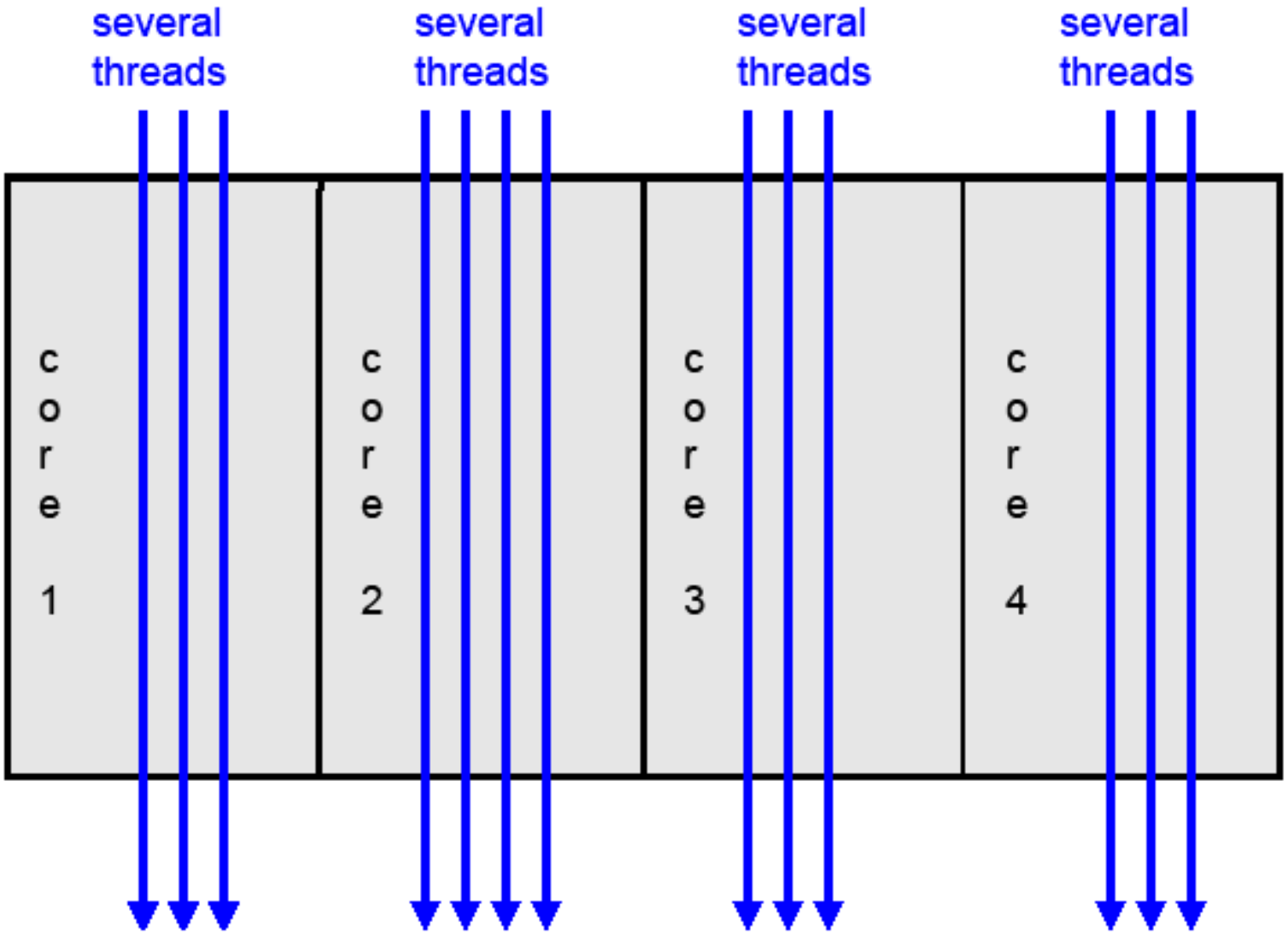
Multi-Core Architecture

- Somewhat recent trend in computer architecture
- Replicate many cores on a single die



Multi-core Chip

Within each core, threads are time-sliced (just like on a uniprocessor)



Interaction With the Operating System

- OS perceives each core as a separate processor
- OS scheduler maps threads/processes to different cores
- Most major OS support multi-core today:
  - Mac OS X, Linux, Windows, …

Today

- Multi-core
- Parallelism on multi-core

Instruction-Level Parallelism

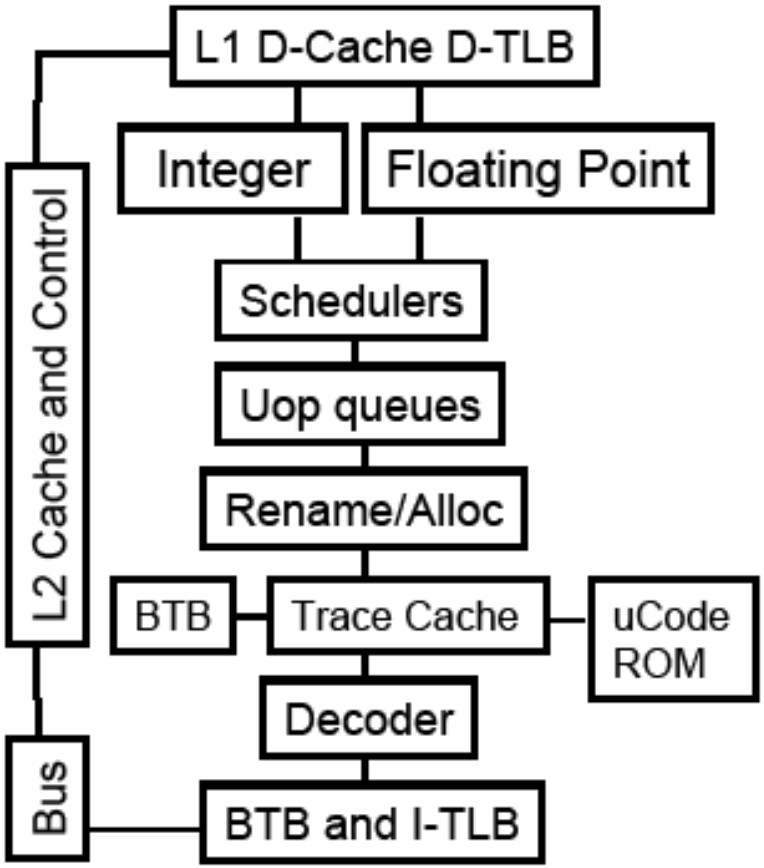
- Parallelism at the machine-instruction level
- Achieved in the processor with
  - Pipeline
  - Re-ordered instructions
  - Split into micro-instructions
  - Aggressive branch prediction
  - Speculative execution
- ILP enabled rapid increases in processor performance
  - Has since plateaued

Thread-level Parallelism

- Parallelism on a coarser scale
- Server can serve each client in a separate thread
  - Web server, database server
- Computer game can do AI, graphics, physics, UI in four different threads
- Single-core superscalar processors cannot fully exploit TLP
  - Thread instructions are interleaved on a coarse level with other threads
- Multi-core architectures are the next step in processor evolution: explicitly exploiting TLP

Simultaneous Multithreading (SMT)

- Complimentary technique to multi-core
- Addresses the stalled pipeline problem
  - Pipeline is stalled waiting for the result of a long operation (float?)
  - … or waiting for data to arrive from memory (long latency)
- Other execution units are idle

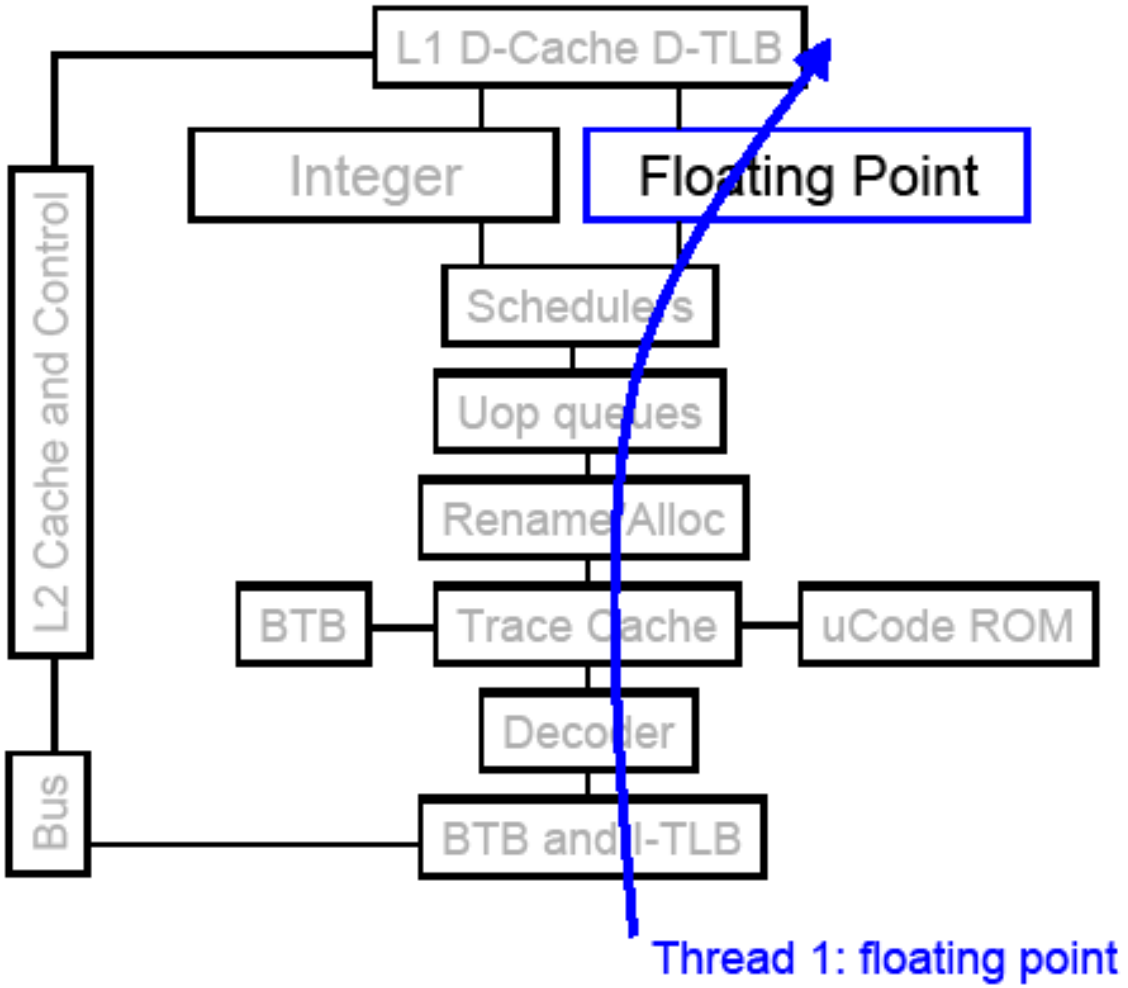


Source: Intel

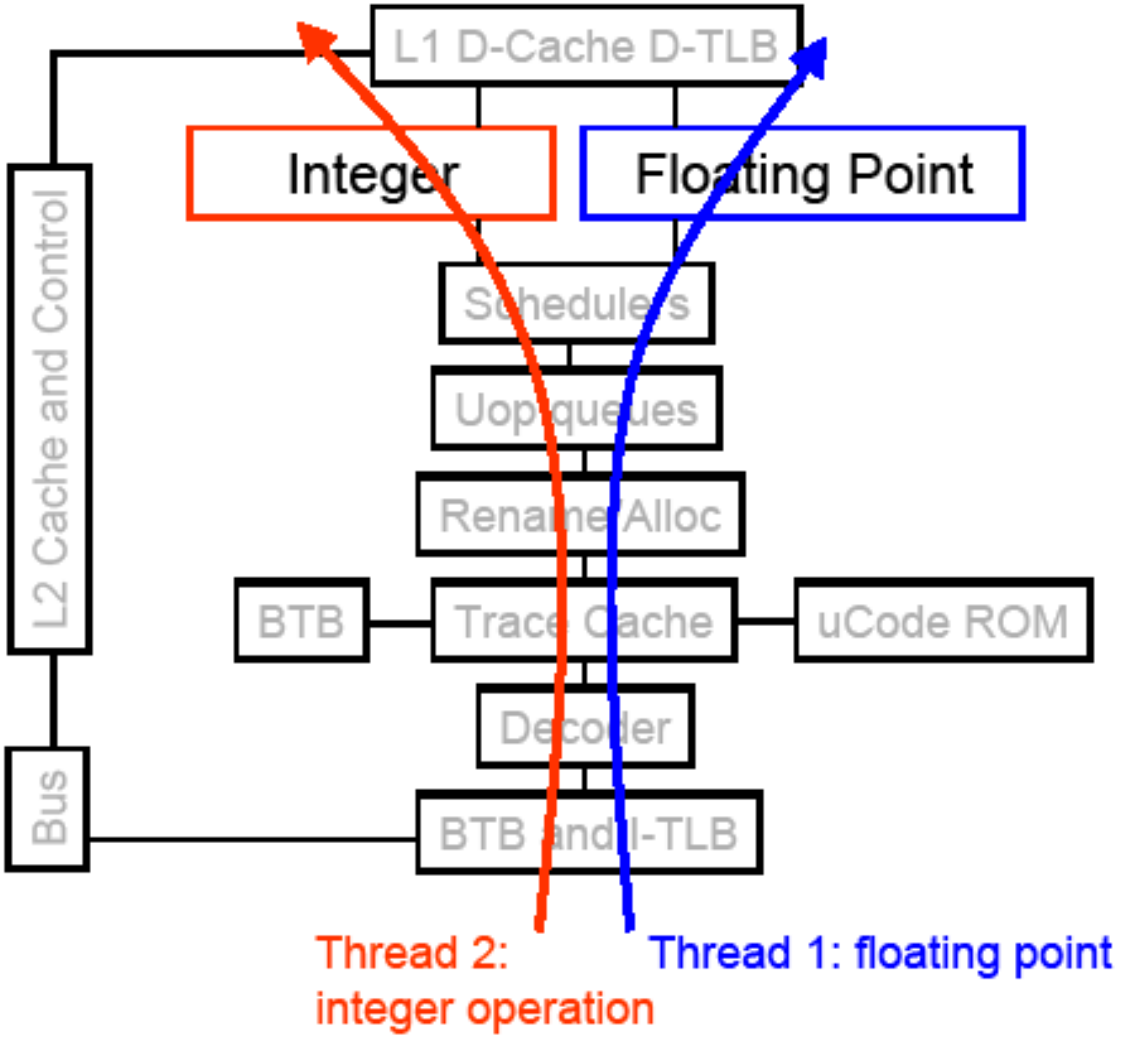
SMT

- Permits multiple independent threads to execute SIMULTANEOUSLY on the SAME core
- Weaving together multiple “threads”
- Example: if one thread is waiting for a floating point operation to complete, another thread can use the integer units

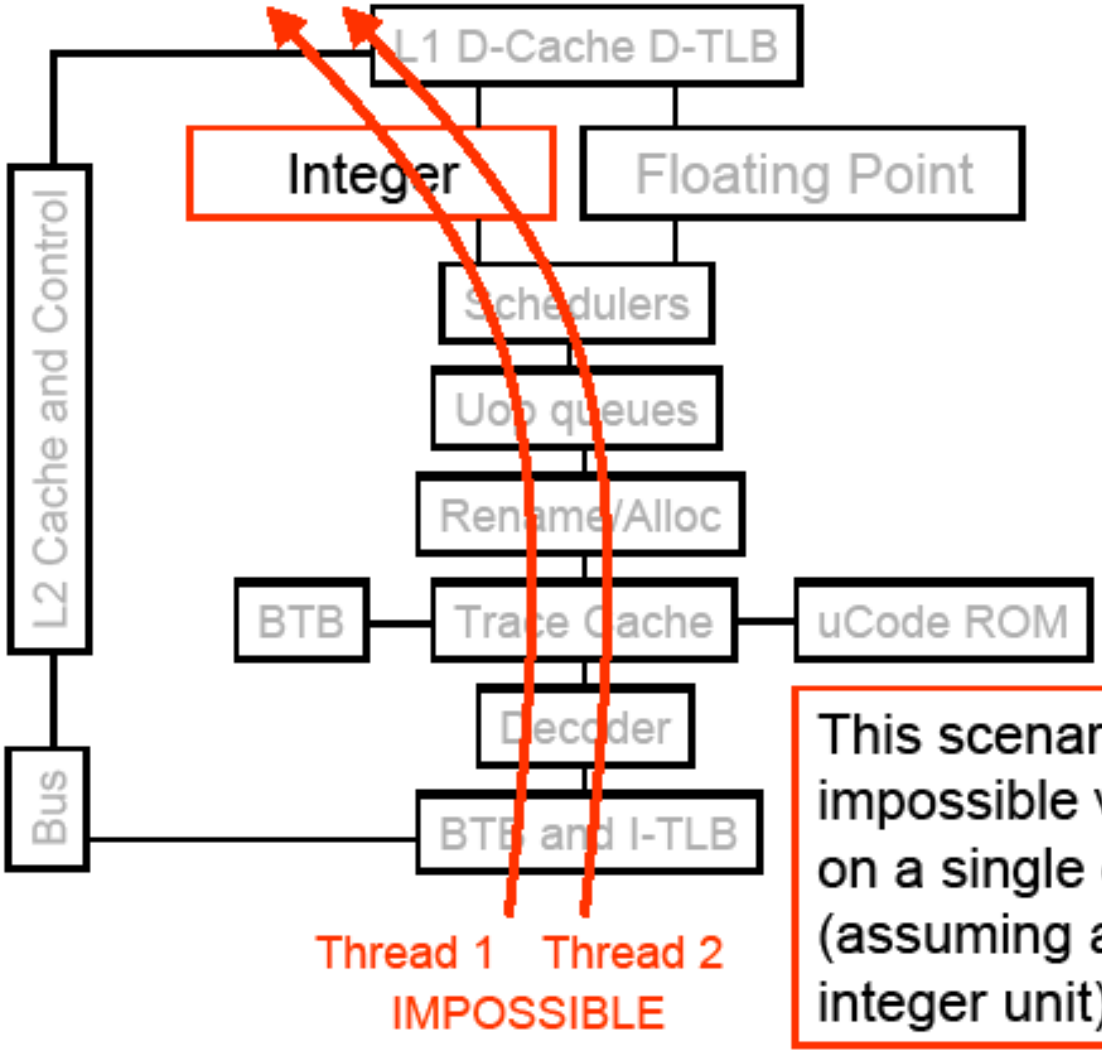
Without SMT, only a single thread can run at any given time



SMT processor: both threads can run concurrently



But: Can't simultaneously use the same functional unit

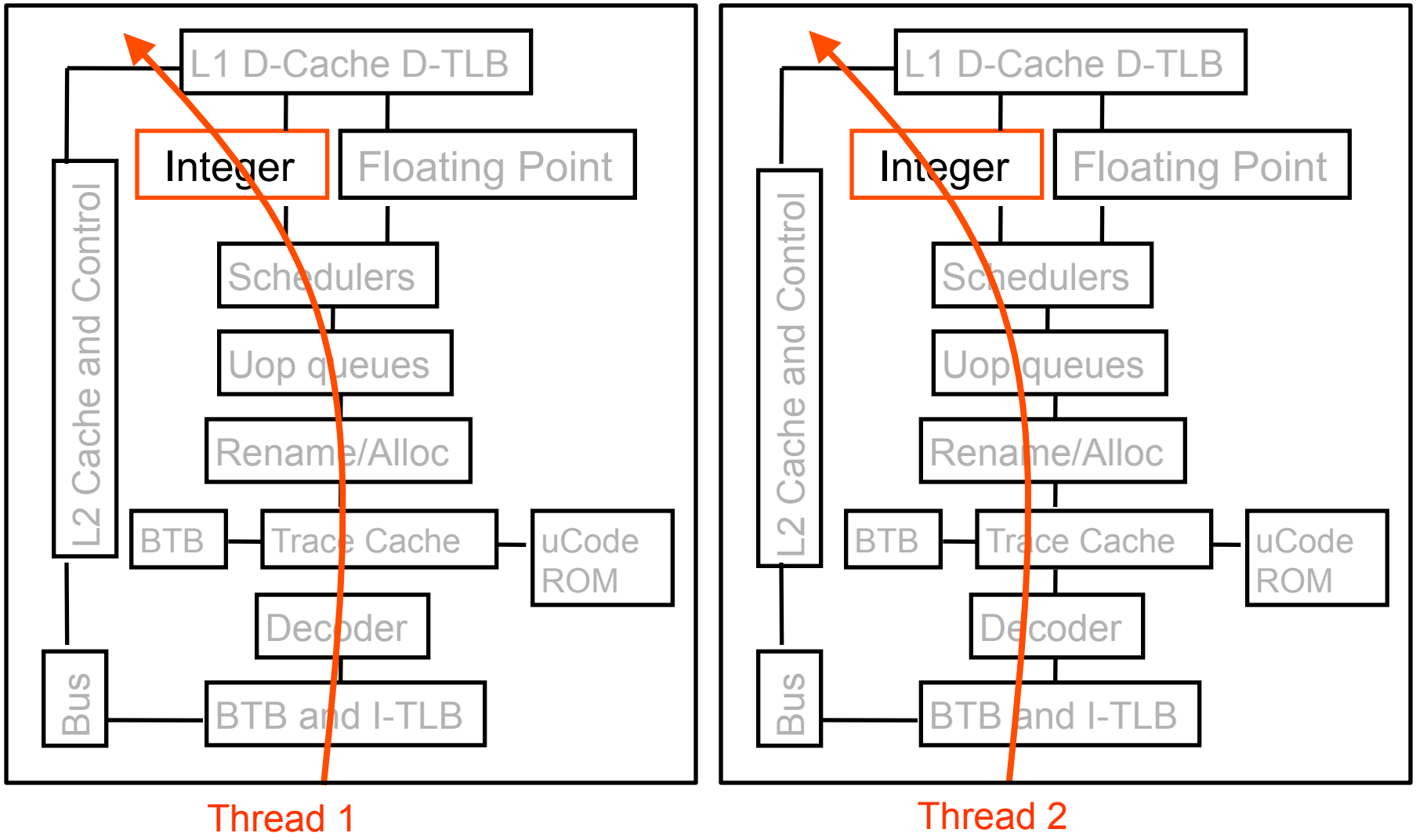


This scenario is impossible with SMT on a single core (assuming a single integer unit)

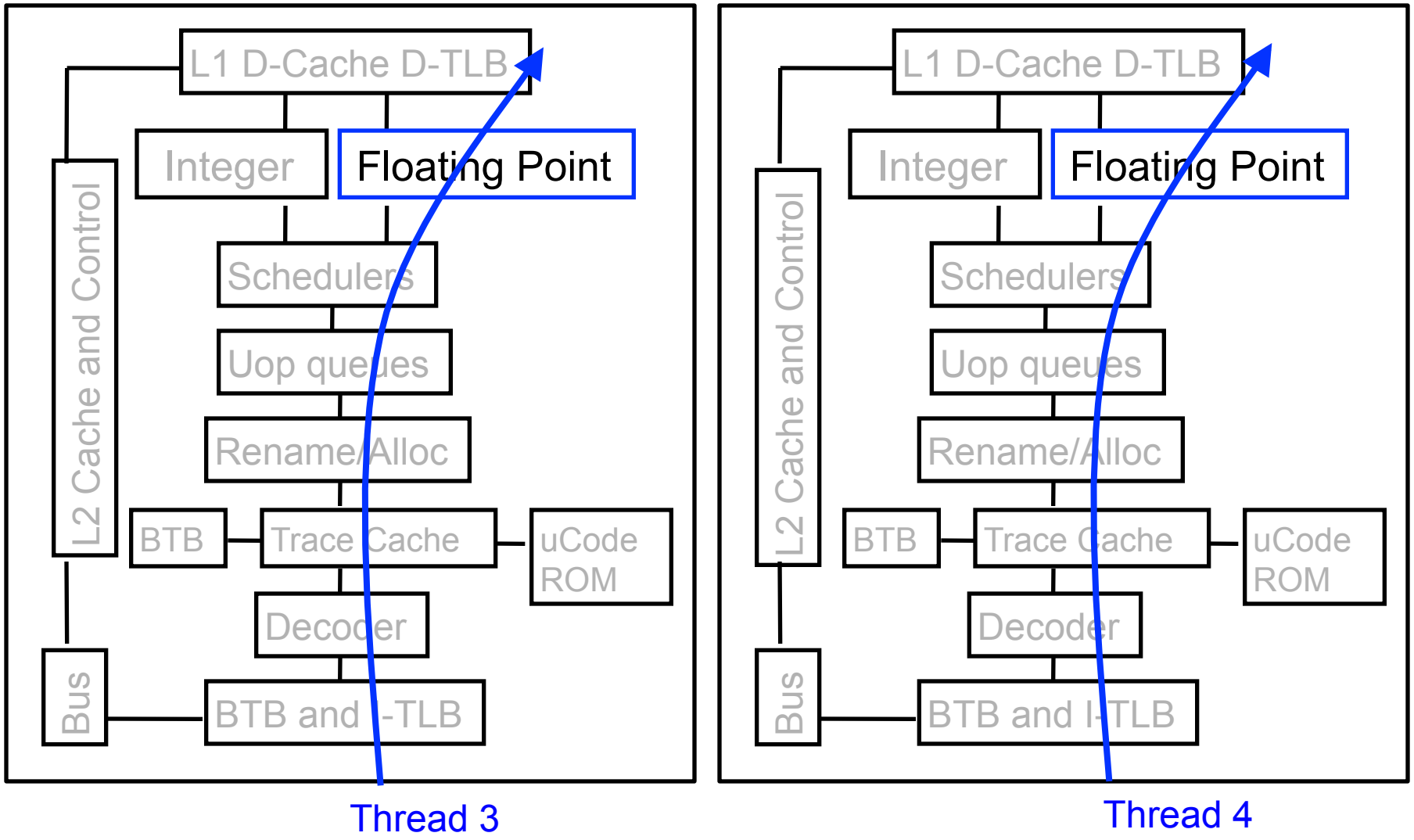
SMT is not a “true” parallel processor

- Enables better threading (e.g. up to 30%)
- OS and applications perceive each simultaneous thread as a separate “virtual processor”
- The chip has only a single copy of each resource
- Compare to multi-core:
  - Each core has its own copy of resources

Multi-core: Threads run on separate cores



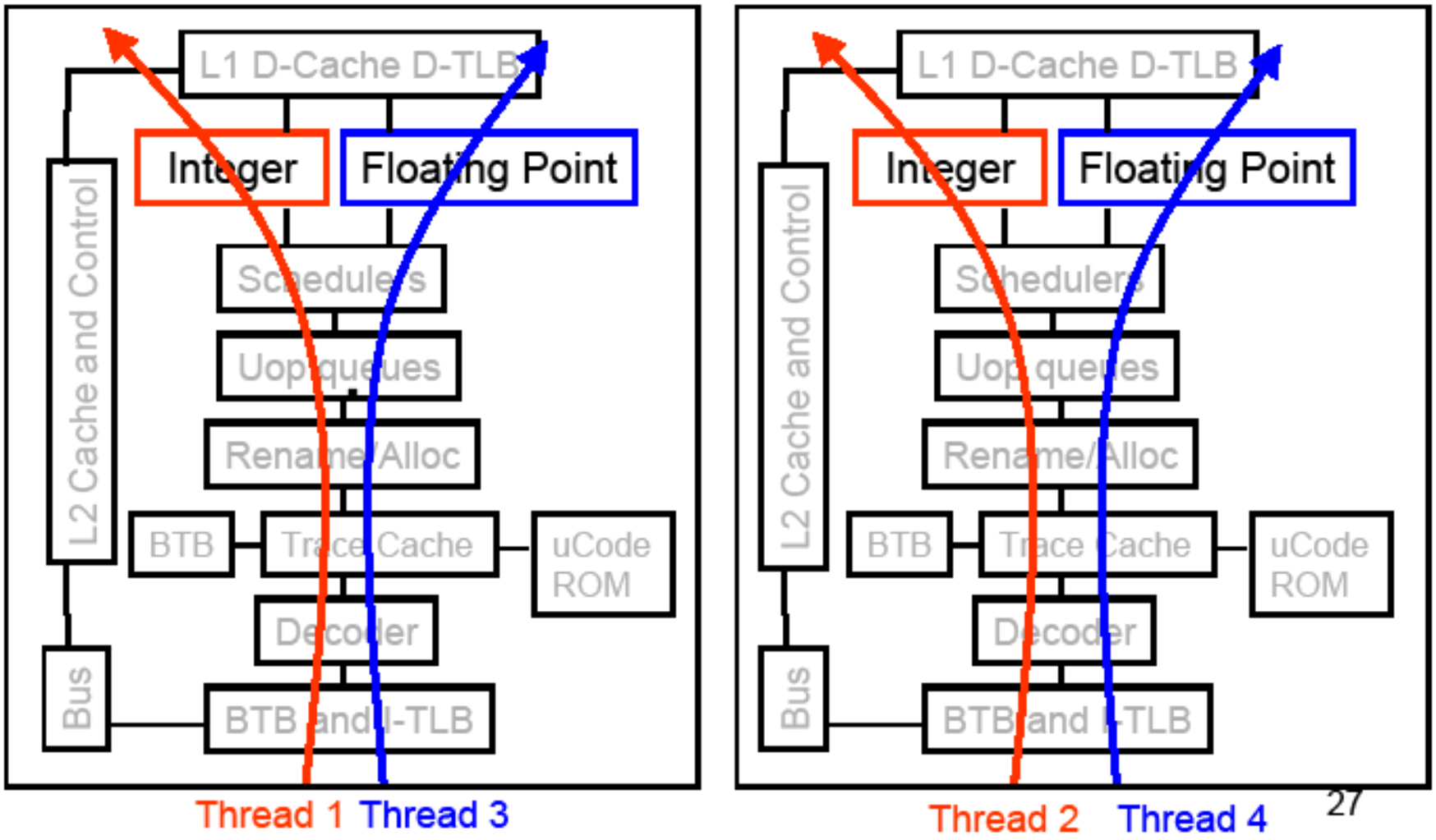
Multi-core: Threads run on separate cores



Combining Multi-core and SMT

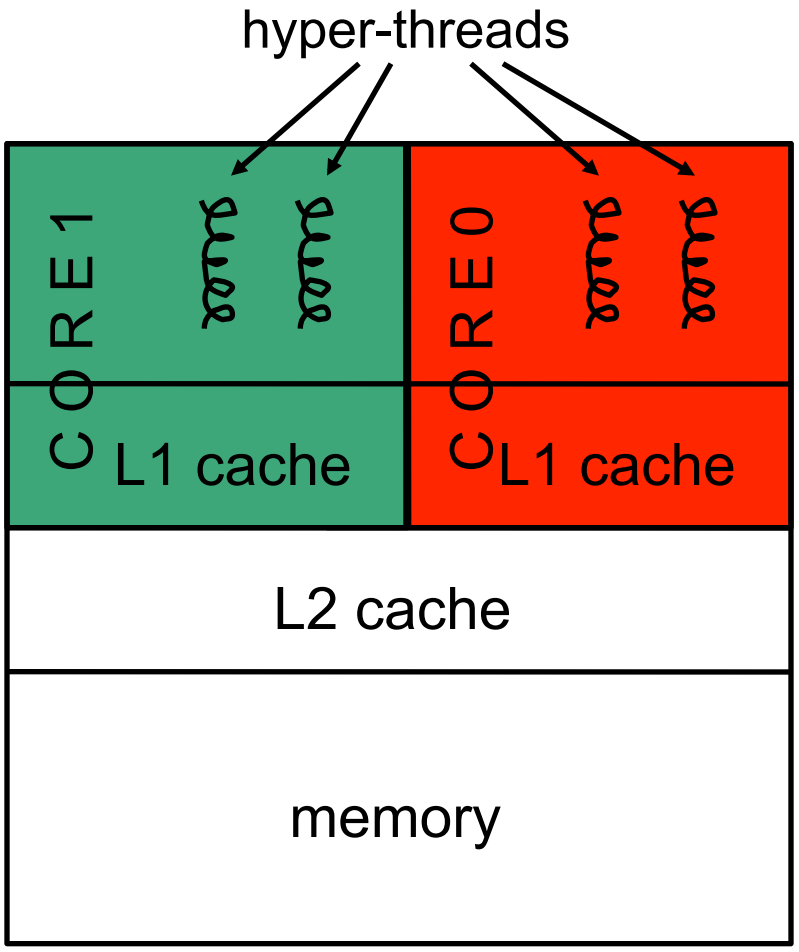
- Cores can be SMT-enabled (or not)
- The different combinations:
  - Single-core, non-SMT: standard uniprocessor
  - Single-core, with SMT
  - Multi-core, non-SMT
  - Multi-core, with SMT: our fish machines
- The number of SMT threads is determined by hardware design
  - 2, 4 or sometimes 8 simultaneous threads
- Intel calls them “Hyper-threads”

SMT Dual-core: all four threads can run concurrently

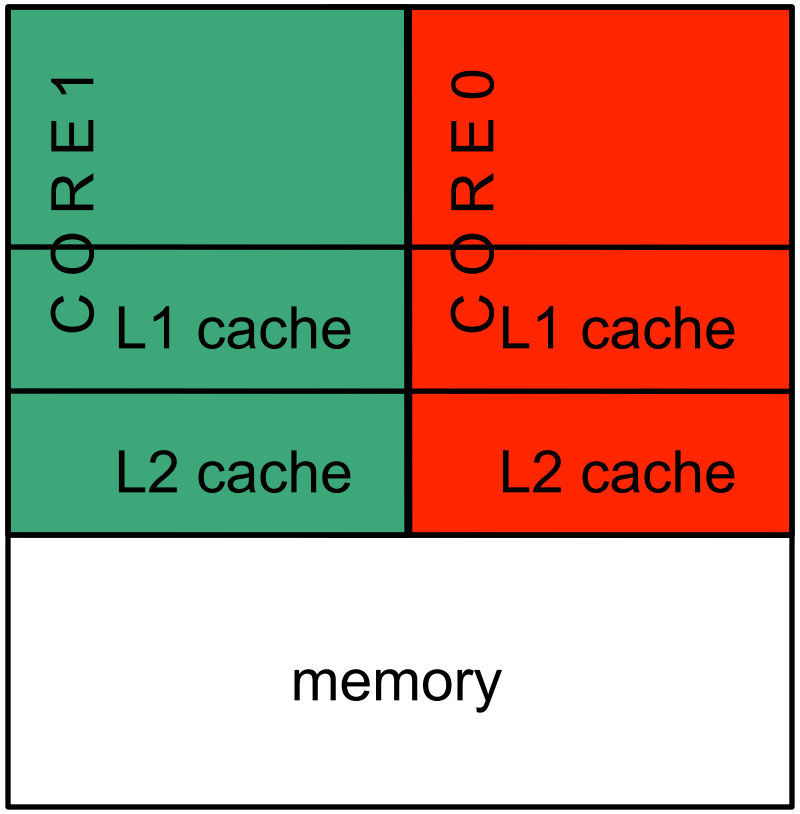


SMT/Multi-Core and the Memory Hierarchy

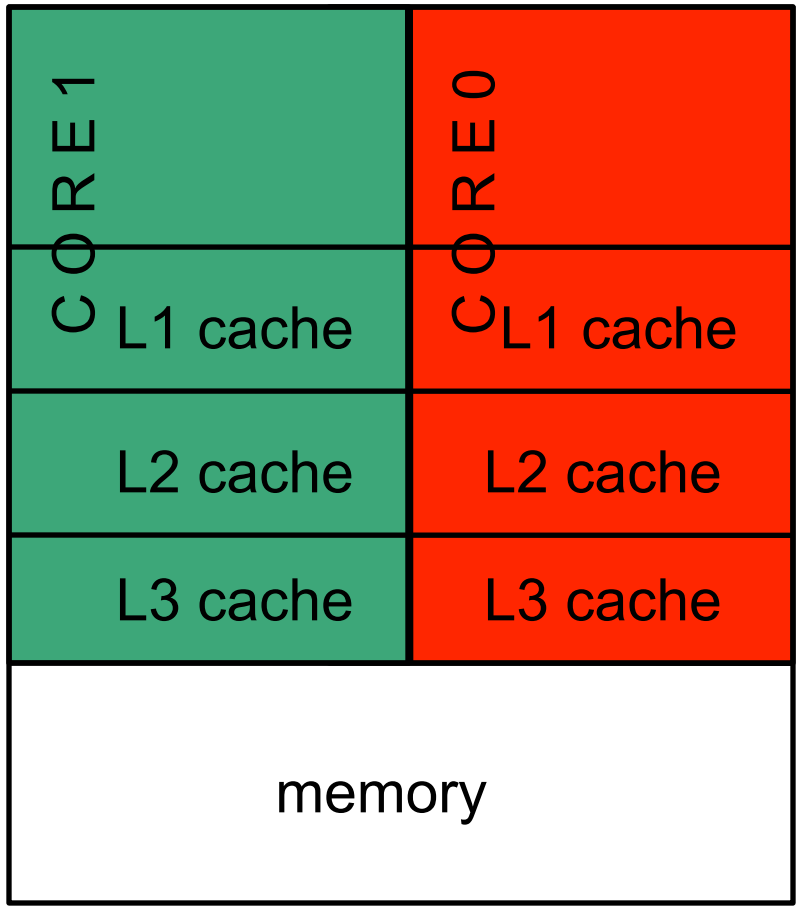
- SMT is a sharing of pipeline resources
  - Thus all caches are shared
- Multi-core chips:
  - L1 caches are private (i.e. each core has its own L1)
  - L2 cache private in some architectures, shared in others
  - Main memory is always shared
- Example: Fish machines
  - Dual-core Intel Xeon processors
  - Each core is hyper-threaded
  - Private L1, shared L2 caches



Designs with Private L2 Caches



Examples: AMD Opteron,
AMD Athlon, Intel Pentium D

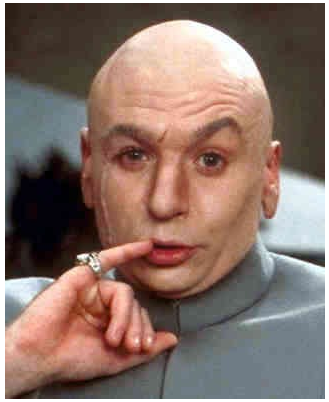


Example: Intel Itanium 2

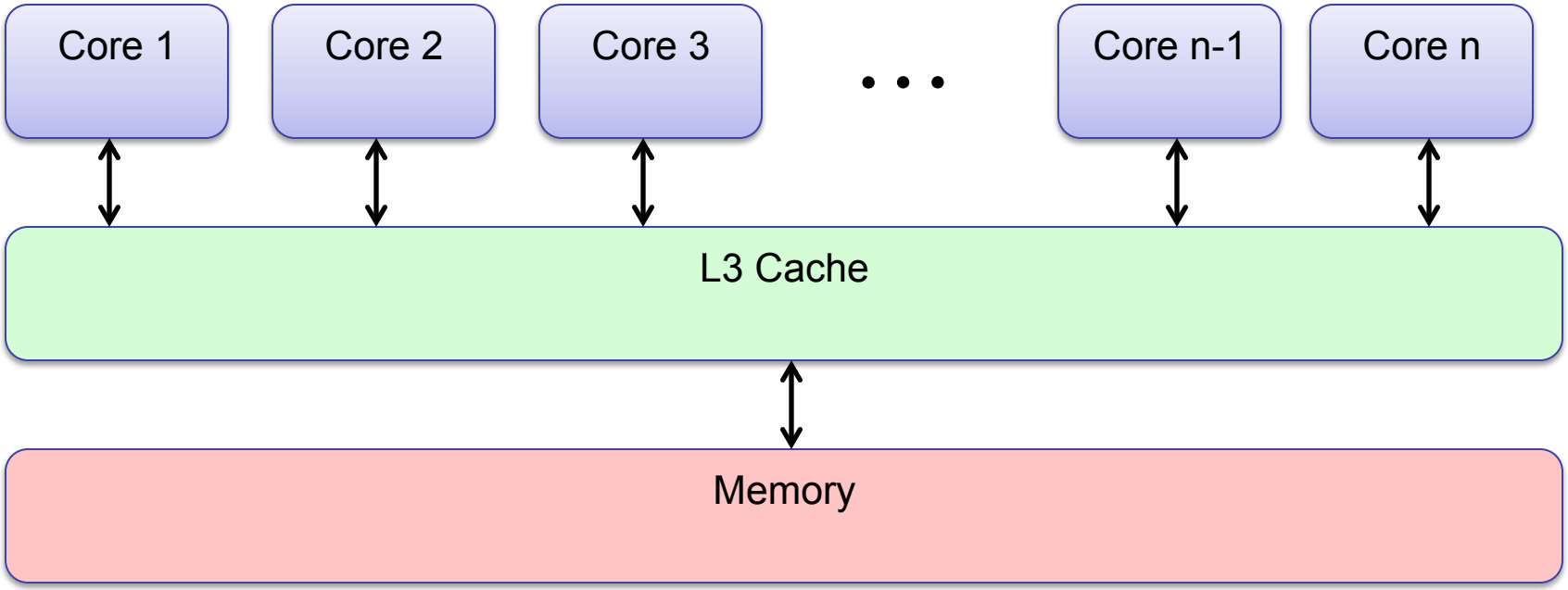
Quad Core 2 Duo shares L2 in pairs of cores

Private vs Shared Cache

- Advantages of Private Cache
  - Closer to the core, so faster access
  - No contention for core access -- no waiting while another core accesses
- Advantages of Shared Cache
  - Threads on different cores can share same cache data
  - More cache space is available if a single (or a few) high-performance threads run
- Cache Coherence Problem
  - The same memory value can be stored in multiple private caches
  - Need to keep the data consistent across the caches
  - Many solutions exist
    - Invalidation protocol with bus snooping, ...

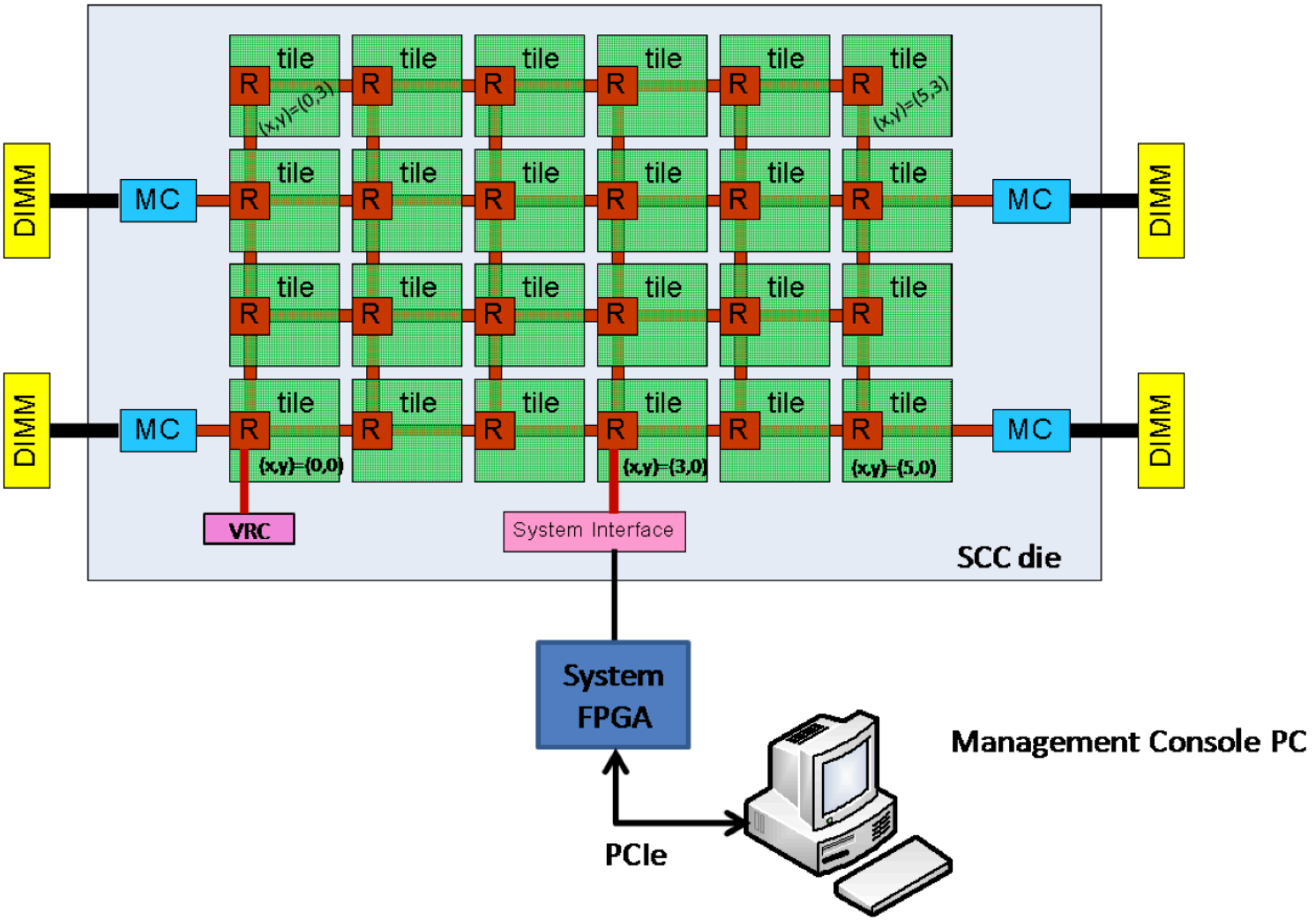


1 MILLION cores!

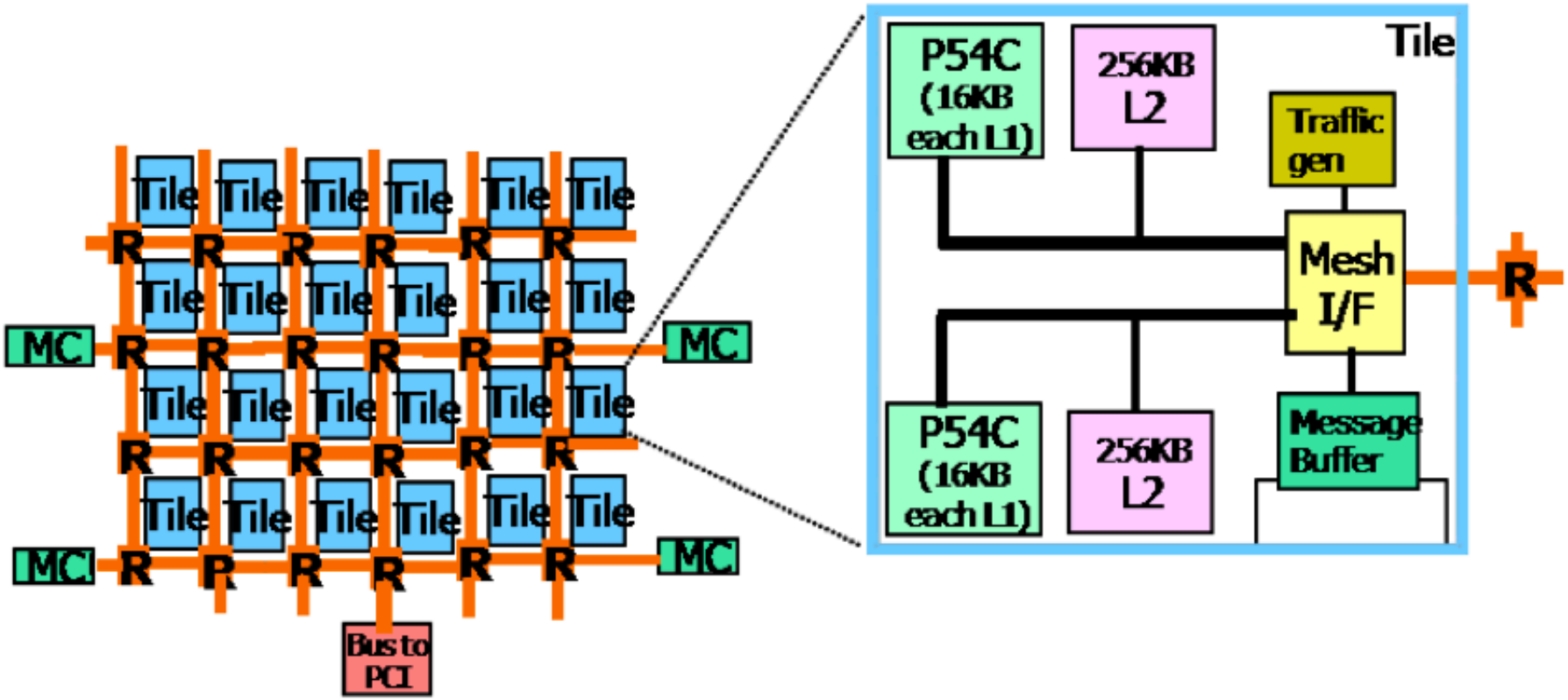


Can this really scale?

Network On A Chip (NOC)



Inside a Tile



Summary

- Multi-Core Architectures
- Simultaneous Multithreading

- **Exam Review Session: WEH 7500 Sunday 6-8pm**
  - **Email Questions to 15213review@gmail.com**

- Next Time:
  - There is no next time ☹