Lecture 12: Cache Innovations

- Today: cache access basics and innovations (Section 2.2)
- TA office hours on Fri 3-4pm
- Tuesday Midterm: open book, open notes, material in first ten lectures (excludes this week)
- Arrive early, 100 mins, 10:35-12:15, manage time well

More Cache Basics

- L1 caches are split as instruction and data; L2 and L3 are unified
- The L1/L2 hierarchy can be inclusive, exclusive, or non-inclusive
- On a write, you can do write-allocate or write-no-allocate
- On a write, you can do writeback or write-through; write-back reduces traffic, write-through simplifies coherence
- Reads get higher priority; writes are usually buffered
- L1 does parallel tag/data access; L2/L3 does serial tag/data

Tolerating Miss Penalty

- Out of order execution: can do other useful work while waiting for the miss – can have multiple cache misses -- cache controller has to keep track of multiple outstanding misses (non-blocking cache)

- Hardware and software prefetching into prefetch buffers – aggressive prefetching can increase contention for buses

Techniques to Reduce Cache Misses

- Victim caches
- Better replacement policies – pseudo-LRU, NRU, DRRIP
- Cache compression

Victim Caches

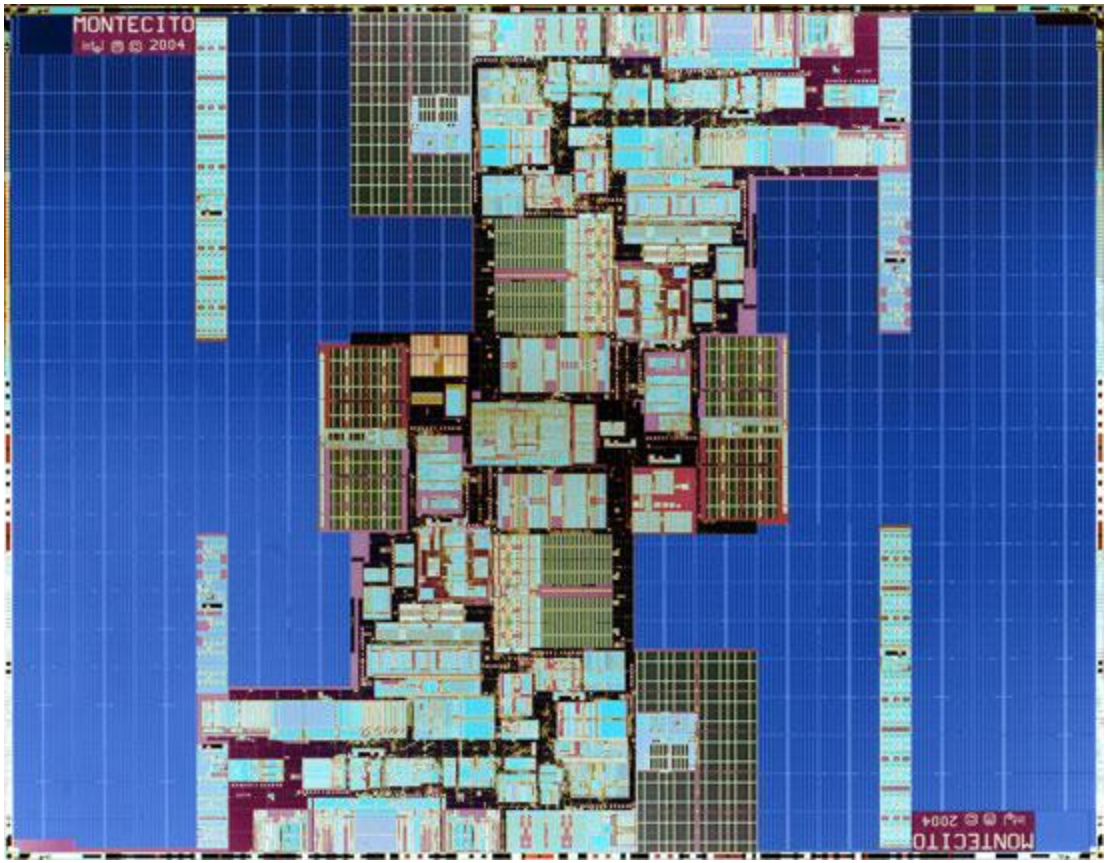
- A direct-mapped cache suffers from misses because multiple pieces of data map to the same location
- The processor often tries to access data that it recently discarded – all discards are placed in a small victim cache (4 or 8 entries) – the victim cache is checked before going to L2
- Can be viewed as additional associativity for a few sets that tend to have the most conflicts

Replacement Policies

- Pseudo-LRU: maintain a tree and keep track of which side of the tree was touched more recently; simple bit ops
- NRU: every block in a set has a bit; the bit is made zero when the block is touched; if all are zero, make all one; a block with bit set to 1 is evicted
- DRRIP: use multiple (say, 3) NRU bits; incoming blocks are set to a high number (say 6), so they are close to being evicted; similar to placing an incoming block near the head of the LRU list instead of near the tail

Intel Montecito Cache

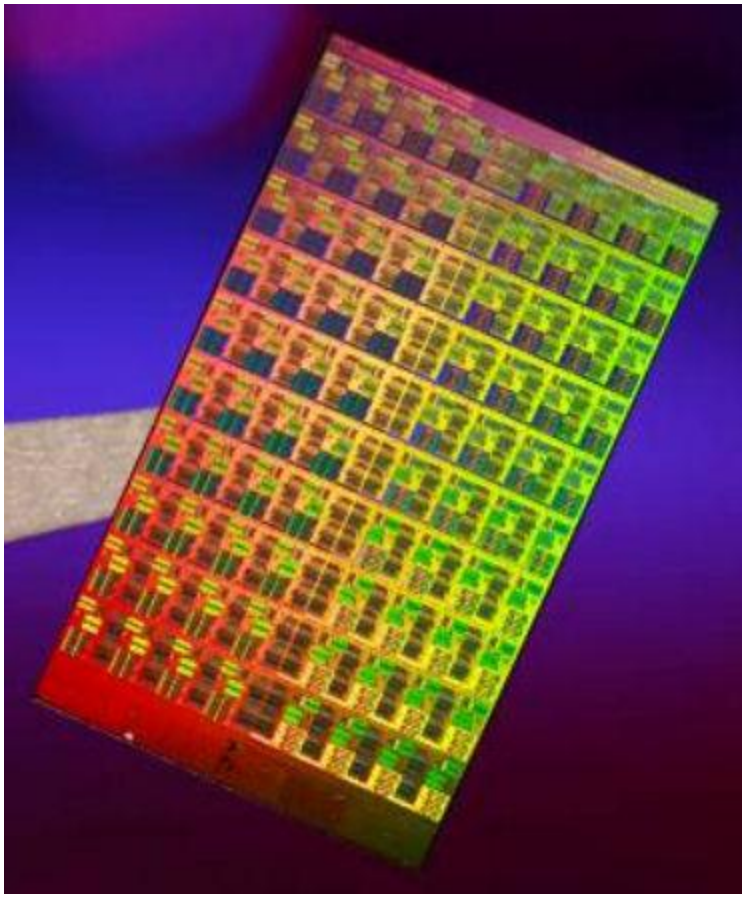
Two cores, each with a private 12 MB L3 cache and 1 MB L2



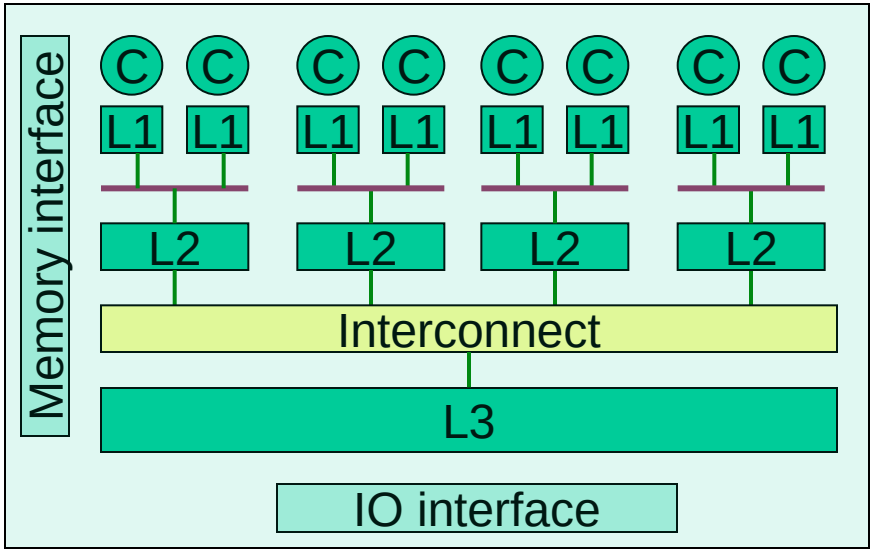
Naffziger et al., Journal of Solid-State Circuits, 2006

Intel 80-Core Prototype – Polaris

Prototype chip with an entire die of SRAM cache stacked upon the cores



Example Intel Studies

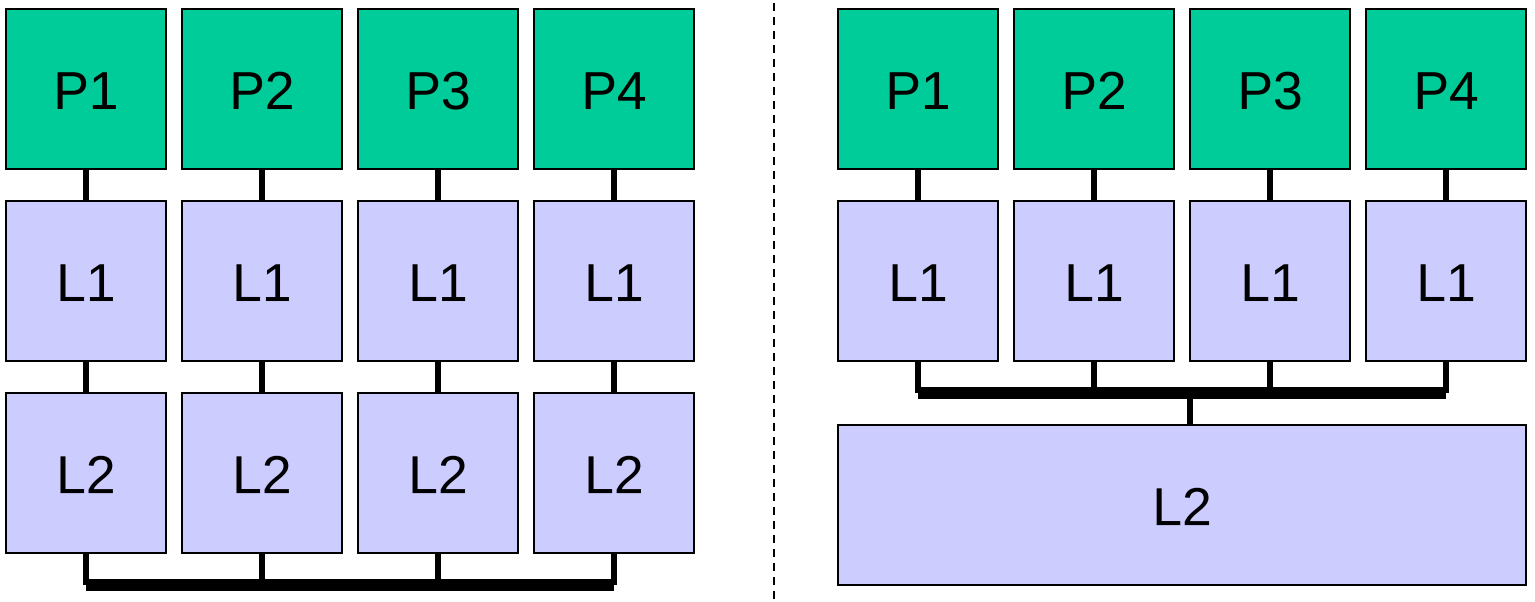


From Zhao et al., CMP-MSI Workshop 2007

L3 Cache sizes up to 32 MB

Shared Vs. Private Caches in Multi-Core

- What are the pros/cons to a shared L2 cache?



Shared Vs. Private Caches in Multi-Core

- Advantages of a shared cache:
  - Space is dynamically allocated among cores
  - No waste of space because of replication
  - Potentially faster cache coherence (and easier to locate data on a miss)

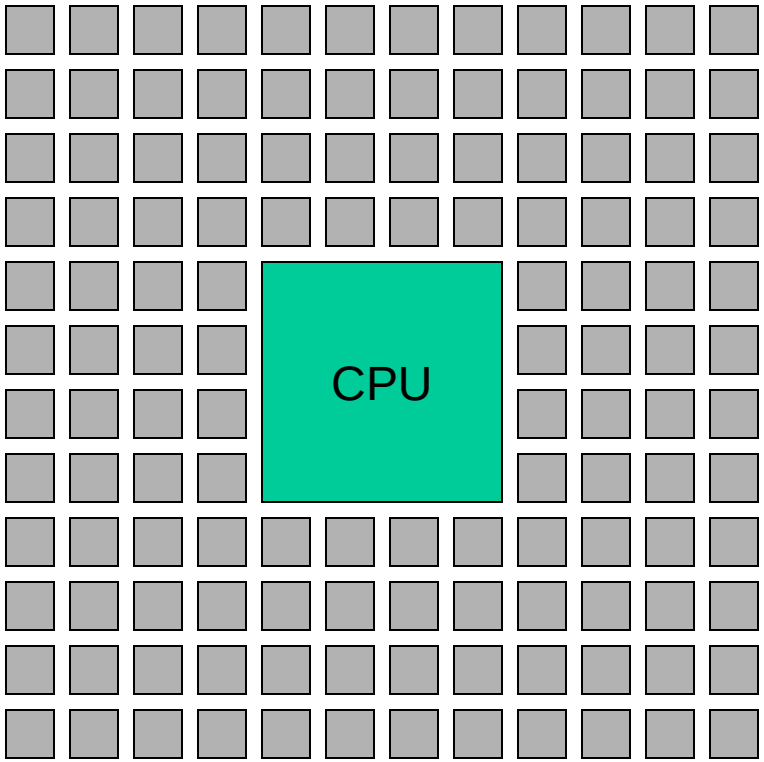
- Advantages of a private cache:
  - small L2 → faster access time
  - private bus to L2 → less contention

UCA and NUCA

- The small-sized caches so far have all been uniform cache access: the latency for any access is a constant, no matter where data is found

- For a large multi-megabyte cache, it is expensive to limit access time by the worst case delay: hence, non-uniform cache architecture

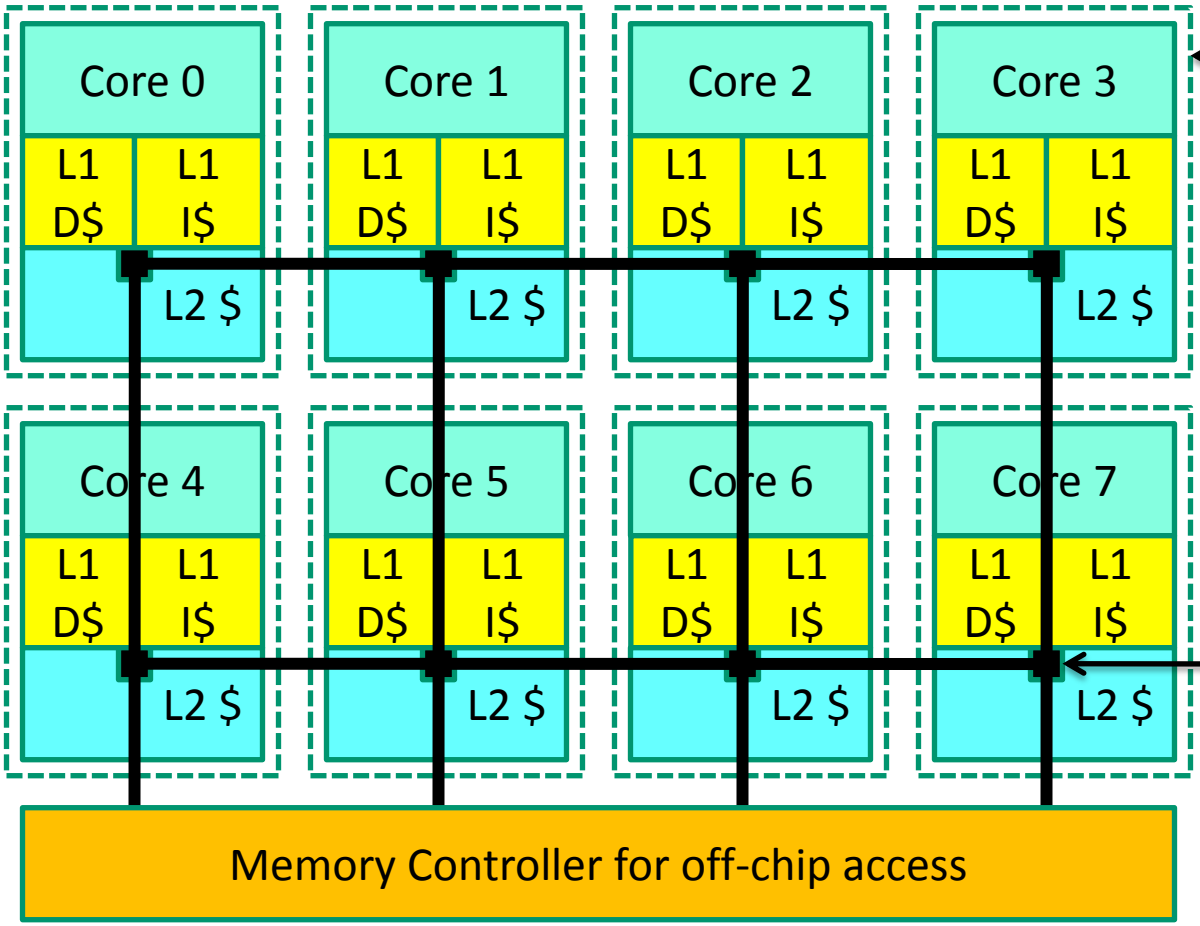
Large NUCA



Issues to be addressed for Non-Uniform Cache Access:

- Mapping
- Migration
- Search
- Replication

Shared NUCA Cache



A single tile composed of a core, L1 caches, and a bank (slice) of the shared L2 cache

The cache controller forwards address requests to the appropriate L2 bank and handles coherence operations

Title

- Bullet