Lecture 1: Introduction

• Course organization:
  ➢ 4 lectures on cache coherence and consistency
  ➢ 2 lectures on transactional memory
  ➢ 2 lectures on interconnection networks
  ➢ 4 lectures on caches
  ➢ 4 lectures on memory systems
  ➢ 4 lectures on core design
  ➢ 2 lectures on parallel algorithms
  ➢ 5 lectures: student paper presentations
  ➢ 2 lectures: student project presentations

• CS 7960-002 for those that want to sign up for 1 credit
Logistics

More Logistics

• Projects: simulation-based, creative, teams of up to 4 students, be prepared to spend time towards middle and end of semester – more details on simulators in a few weeks

• Final project report due in late April (will undergo conference-style peer reviewing); also watch out for workshop deadlines for ISCA

• Grading:
  - 70% project
  - 10% paper presentation
  - 20% take-home final
Multi-Core Cache Organizations

Private L1 caches
Shared L2 cache
Bus between L1s and single L2 cache controller
Snooping-based coherence between L1s
Multi-Core Cache Organizations

Private L1 caches
Shared L2 cache, but physically distributed
Scalable network
Directory-based coherence between L1s
Multi-Core Cache Organizations

- Private L1 caches
- Shared L2 cache, but physically distributed
- Bus connecting the four L1s and four L2 banks
- Snooping-based coherence between L1s
Multi-Core Cache Organizations

Private L1 caches
Private L2 caches
Scalable network
Directory-based coherence between L2s (through a separate directory)
Shared-Memory Vs. Message Passing

• Shared-memory
  ▪ single copy of (shared) data in memory
  ▪ threads communicate by reading/writing to a shared location

• Message-passing
  ▪ each thread has a copy of data in its own private memory that other threads cannot access
  ▪ threads communicate by passing values with SEND/RECEIVE message pairs
Cache Coherence

A multiprocessor system is cache coherent if

• a value written by a processor is eventually visible to reads by other processors – write propagation

• two writes to the same location by two processors are seen in the same order by all processors – write serialization
Cache Coherence Protocols

• Directory-based: A single location (directory) keeps track of the sharing status of a block of memory

• Snooping: Every cache block is accompanied by the sharing status of that block – all cache controllers monitor the shared bus so they can update the sharing status of the block, if necessary

  ➢ Write-invalidate: a processor gains exclusive access of a block before writing by invalidating all other copies
  ➢ Write-update: when a processor writes, it updates other shared copies of that block
Protocol-I  MSI

- 3-state write-back invalidation bus-based snooping protocol
- Each block can be in one of three states – invalid, shared, modified (exclusive)
- A processor must acquire the block in exclusive state in order to write to it – this is done by placing an exclusive read request on the bus – every other cached copy is invalidated
- When some other processor tries to read an exclusive block, the block is demoted to shared
Design Issues, Optimizations

• When does memory get updated?
  ➢ demotion from modified to shared?
  ➢ move from modified in one cache to modified in another?

• Who responds with data? – memory or a cache that has the block in exclusive state – does it help if sharers respond?

• We can assume that bus, memory, and cache state transactions are atomic – if not, we will need more states

• A transition from shared to modified only requires an upgrade request and no transfer of data
Reporting Snoop Results

• In a multiprocessor, memory has to wait for the snoop result before it chooses to respond – need 3 wired-OR signals: (i) indicates that a cache has a copy, (ii) indicates that a cache has a modified copy, (iii) indicates that the snoop has not completed

• Ensuring timely snoops: the time to respond could be fixed or variable (with the third wired-OR signal)

• Tags are usually duplicated if they are frequently accessed by the processor (regular ld/sts) and the bus (snoops)
4 and 5 State Protocols

• Multiprocessors execute many single-threaded programs

• A read followed by a write will generate bus transactions to acquire the block in exclusive state even though there are no sharers (leads to MESI protocol)

• Also, to promote cache-to-cache sharing, a cache must be designated as the responder (leads to MOESI protocol)

• Note that we can optimize protocols by adding more states – increases design/verification complexity
MERSI Protocol

• The new state is exclusive-clean – the cache can service read requests and no other cache has the same block.

• When the processor attempts a write, the block is upgraded to exclusive-modified without generating a bus transaction.

• When a processor makes a read request, it must detect if it has the only cached copy – the interconnect must include an additional signal that is asserted by each cache if it has a valid copy of the block.

• When a block is evicted, a block may be exclusive-clean, but will not realize it.
MOESI Protocol

- The first reader or the last writer are usually designated as the owner of a block.
- The owner is responsible for responding to requests from other caches.
- There is no need to update memory when a block transitions from M → S state.
- The block in O state is responsible for writing back a dirty block when it is evicted.
Title

• Bullet