Cache Coherence

Cache coherent processors
- reading processor must get the most current value
- most current value is the last write

Cache coherence problem
- updates from 1 processor not known to others

Mechanisms for maintaining cache coherence
- cache coherence protocols
- coherency state associated with a block of data
- bus/interconnect operations on shared data change the state
  - for the processor that initiates an operation
  - for other processors that have the data of the operation resident in their caches
Cache Coherency Protocols

Write-invalidate
(Sequent Symmetry, SGI Power/Challenge, SPARCCenter 2000)
- processor obtains exclusive access for writes (becomes the
  "owner") by invalidating data in other processors' caches
- coherency miss (invalidation miss)
- cache-to-cache transfers
- good for:
  - multiple writes to same word or block by one processor
  - migratory sharing from processor to processor
- a problem is false sharing
  - processors read & write to different words in a shared cache
    block
  - cache coherency is maintained on a cache block basis
    - block ownership bounces between processor caches

A Low-end MP

- Processor
  - One or more levels of cache
- Main memory

- Processor
  - One or more levels of cache
- I/O System

- Processor
  - One or more levels of cache

- Processor
  - One or more levels of cache
Cache Coherency Protocols

Write-update
(SPARCCenter 2000)
- broadcast each write to actively shared data
- each processor with a copy snarfs the data
- good for inter-processor contention

Competitive
- switches between them

We will focus on write-invalidate.
Cache Coherency Protocol Implementations

**Snooping**
- used with low-end MPs
  - few processors
  - centralized memory
  - bus-based

**Directory-based**
- used with higher-end MPs
  - more processors
  - distributed memory
  - multipath interconnect
**Snooping Implementation**

A distributed coherency protocol
- coherency state associated with each cache block
- each snoop maintains coherency for its own cache

How the bus is used
- broadcast medium
- entire coherency operation is atomic wrt other processors
  - *keep-the-bus protocol*: master holds the bus until the entire operation has completed
  - *split-transaction buses*:
    - request & response are different phases
    - state value that indicates that an operation is in progress
    - do not initiate another operation for a cache block that has one in progress

**Snooping Implementation**

Snoop implementation:
- separate tags & state for snoop lookups
  - processor & snoop communicate for a state or tag change
- snoop on the highest level cache
  - another reason it is a physically-accessed cache
  - property of *inclusion*:
    - all blocks in L1 are in L2
    - therefore only have to snoop on L2
    - may need to update L1 state if change L2 state
An Example Snooping Protocol

**Invalidation-based** coherency protocol
Each cache block is in one of three states
- **shared:**
  - clean in all caches & up-to-date in memory
  - block can be read by any processor
- **exclusive:**
  - dirty in exactly one cache
  - only that processor can write to it
- **invalid:**
  - block contains no valid data

State Transitions for a Given Cache Block

State transitions caused by:
- events incurred by the **processor associated with the cache**
  - read miss, write miss, write on shared block
- events incurred by **snooping on the bus** as a result of other processor actions, e.g.,
  - read miss by P1 makes P2's block change from exclusive to shared
  - write miss by P1 makes P2's block change from exclusive to invalid
**State Machine (CPU side)**

- **Invalid**
  - CPU read miss
  - Place read op on bus
  - CPU write miss
  - Place write op on bus
  - CPU read hit

- **Exclusive** (read/write)
  - CPU read miss
  - Place read op on bus
  - CPU write miss
  - Place write op on bus
  - CPU write hit

- **Shared** (read/only)
  - CPU read miss
  - Place read op on bus
  - CPU write
  - Place write op on bus

**State Machine (Bus side: the snoop)**

- **Invalid**
  - Write miss for this block
  - Write back the block

- **Exclusive** (read/write)
  - Write miss for this block
  - Write back the block

- **Shared** (read/only)
  - Read miss for this block
  - Write back the block