CMSC 411
Computer Systems Architecture
Lecture 22
Multiprocessors, cont.

Alan Sussman
als@cs.umd.edu

Administrivia

- Exam #2 returned Thursday
- Project #2 cache simulator due tomorrow
  - questions?
- Homework #5 on storage systems posted
  - due in class Tuesday, May 6
- Continue reading Chapter 5

Cache Resources for WB Snooping

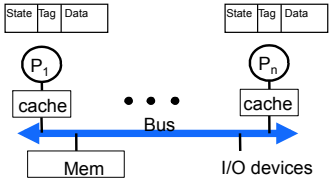
- To track whether a cache block is shared, add extra state bit associated with each cache block, like valid bit and dirty bit, shared bit
  - Write to Shared block ⇒ Need to place invalidate on bus and mark cache block as private (if an option)
  - No further invalidations will be sent for that block
  - This processor called owner of cache block
  - Owner then changes state from shared to unshared (or *exclusive*)

Example Snooping Protocol

- Snooping coherence protocol is usually implemented by incorporating a finite-state controller in each node (cache)
- Logically, think of a separate controller associated with each cache block
  - That is, snooping operations or cache requests for different blocks can proceed independently
- In real implementations, a single controller allows multiple operations to distinct blocks to proceed in interleaved fashion
  - meaning one operation may be initiated before another is completed, even though only one cache access or one bus access is allowed at a time

Write-through Invalidate Protocol

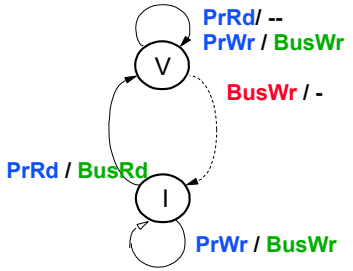
- 2 states per block in each cache
  - as in uniprocessor
  - state of a block is a *p*-vector of states
  - Hardware state bits associated with blocks that are in the cache
  - other blocks can be seen as being in invalid (not-present) state in that cache
- Writes invalidate all other cache copies
  - can have multiple simultaneous readers of block, but write invalidates them



The Example

Format: event / action

PrRd: Processor Read
PrWr: Processor Write
BusRd: Bus Read
BusWr: Bus Write



Single-writer, multiple-reader (SWMR)

Is Example 2-state Protocol Coherent?

Memory is *coherent* iff:

1. A read by P to location X that *follows* a write by P to X, w/ no intervening writes, returns written value.

2. A read by P to location X that *follows* a write by Q to X, w/ no intervening writes, *and the read and write sufficiently separated*, returns written value.

3. Writes to same location are serialized
   - writes to same location by distinct processors seen in same order by all other processors

Is Example 2-state Protocol Coherent?

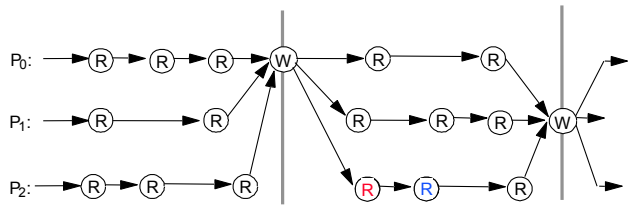
Assume bus transactions and memory ops atomic, and a one-level cache
- all phases of one bus transaction complete before next one starts
- processor waits for memory operation to complete before issuing next
- with one-level cache, assume invalidations applied during bus transaction

Processors only observe state of memory through reads….

Writes only observable by other processors if on bus…

- *All writes go to bus*! (in this example protocol, not all others)
  - Writes serialized by order in which they appear on bus (bus order)
  - invalidations applied to caches in bus order
- How to insert reads in this order?
  - Important since processors see writes through reads, so determines whether write serialization is satisfied
  - But read hits may happen independently and do not appear on bus or enter directly in bus order

Ordering



- Writes establish a partial order
- Doesn't constrain ordering of reads, though shared-medium (bus) will order read misses too
  - any order among reads between writes is fine
- *Writes serialized, reads and writes not interchanged, so coherent!*

Outline

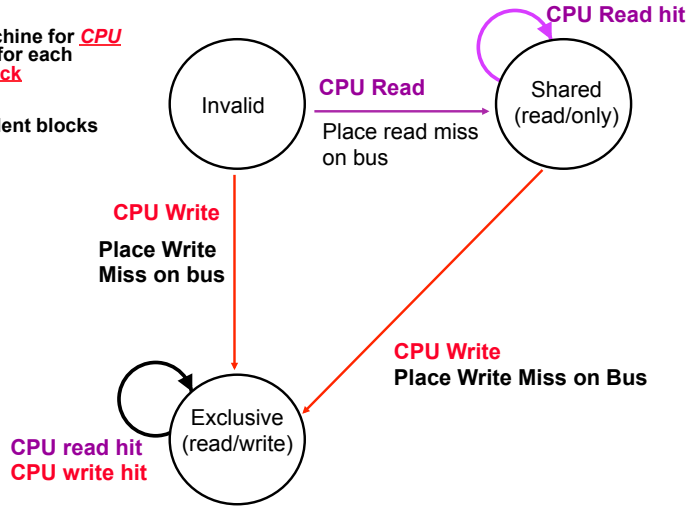
- Review
- Coherence
- Write Consistency
- Administrivia
- Snooping
- Building Blocks
- Snooping protocols and examples
- Coherence traffic and Performance on MP
- Directory-based protocols and examples
- Conclusion

Example Write Back Snoopy Protocol

- Invalidation protocol, write-back cache
  - Snoops every address on bus
  - If it has a dirty copy of requested block, provides that block in response to the read request and aborts the memory access
- Each memory block is in one state:
  - Clean in all caches and up-to-date in memory (Shared)
  - OR Dirty in exactly one cache (Exclusive)
  - OR Not in any caches
- Each cache block is in one state (track these):
  - Shared : block can be read
  - OR Exclusive : cache has only copy, its writeable, and dirty
  - OR Invalid : block contains no data (in uniprocessor cache too)
- Read misses: cause all caches to snoop bus
- Writes to clean blocks are treated as misses
  - *write-allocate*

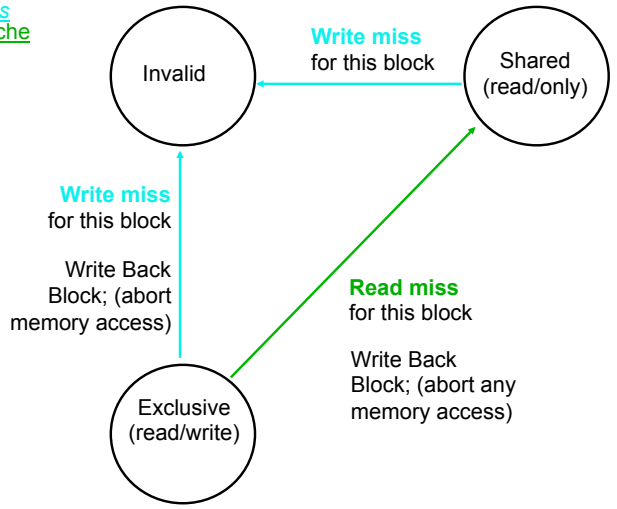
Write-Back State Machine – CPU Events

- State machine for *CPU* requests for each cache block
- Non-resident blocks invalid

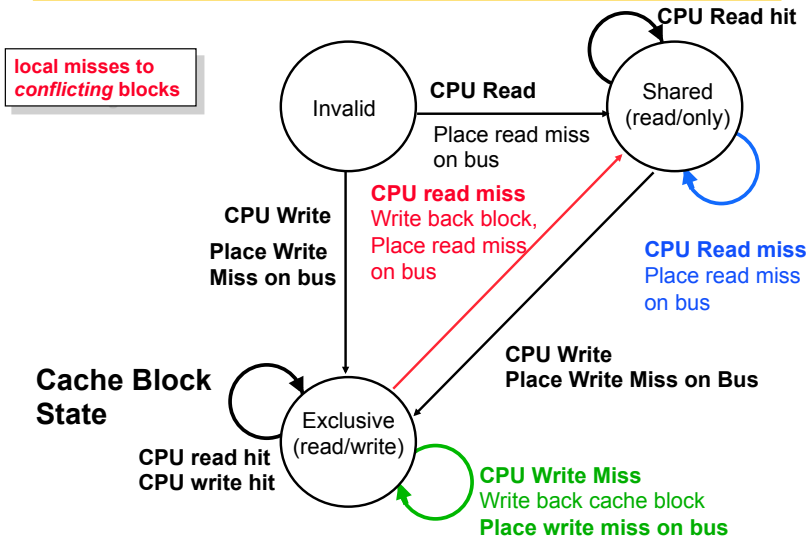


Write-Back State Machine- Bus events

State machine for *bus* requests for each cache block

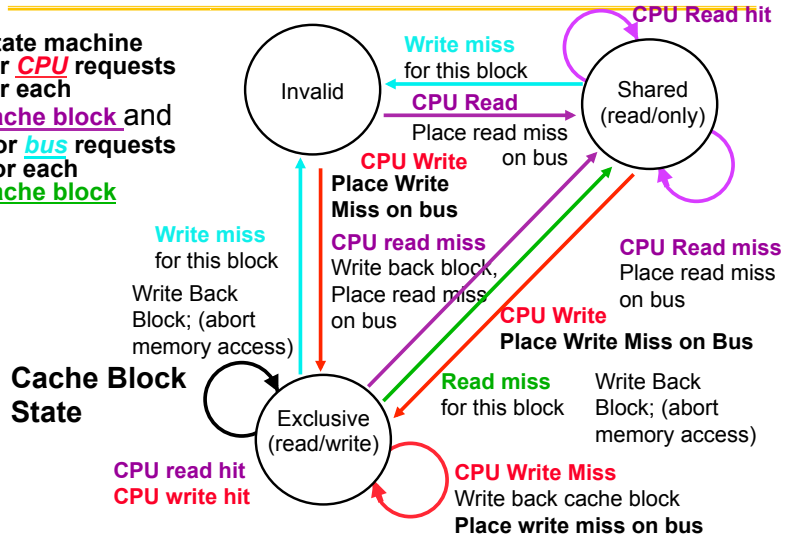


Block-replacement



Write-back State Machine-III

- State machine for *CPU* requests for each cache block and for *bus* requests for each cache block



Example

	P1			P2			Bus				Memory	
step	State	Addr	Value	State	Addr	Value	Action	Proc.	Addr	Value	Addr	Value
P1 Write 10 to A1											A1	0
P1: Read A1												
P2: Read A1												
P2: Write 20 to A1												
P2: Write 40 to A2												

Assumes A1 and A2 map to same cache block, initial cache state is invalid

Example

	P1			P2			Bus				Memory	
step	State	Addr	Value	State	Addr	Value	Action	Proc.	Addr	Value	Addr	Value
P1 Write 10 to A1	Excl.	A1	10				WrMs	P1	A1		A1	0
P1: Read A1												
P2: Read A1												
P2: Write 20 to A1												
P2: Write 40 to A2												

Assumes A1 and A2 map to same cache block

Example

	P1			P2			Bus				Memory	
step	State	Addr	Value	State	Addr	Value	Action	Proc.	Addr	Value	Addr	Value
P1 Write 10 to A1	Excl.	A1	10				WrMs	P1	A1		A1	0
P1: Read A1	Excl.	A1	10								A1	0
P2: Read A1												
P2: Write 20 to A1												
P2: Write 40 to A2												

Assumes A1 and A2 map to same cache block

Example

	P1			P2			Bus				Memory	
step	State	Addr	Value	State	Addr	Value	Action	Proc.	Addr	Value	Addr	Value
P1 Write 10 to A1	Excl.	A1	10				WrMs	P1	A1		A1	0
P1: Read A1	Excl.	A1	10								A1	0
P2: Read A1				Shar.	A1		RdMs	P2	A1		A1	0
	Shar.	A1	10				WrBk	P1	A1	10	A1	10
				Shar.	A1	10	RdDa	P2	A1	10	A1	10
P2: Write 20 to A1												
P2: Write 40 to A2												

Assumes A1 and A2 map to same cache block

Example

	P1			P2			Bus				Memory	
step	State	Addr	Value	State	Addr	Value	Action	Proc.	Addr	Value	Addr	Value
P1 Write 10 to A1	Excl.	A1	10				WrMs	P1	A1		A1	0
P1: Read A1	Excl.	A1	10								A1	0
P2: Read A1				Shar.	A1		RdMs	P2	A1		A1	0
	Shar.	A1	10				WrBk	P1	A1	10	A1	10
				Shar.	A1	10	RdDa	P2	A1	10	A1	10
P2: Write 20 to A1	Inv.			Excl.	A1	20	WrMs	P2	A1		A1	10
P2: Write 40 to A2												

Assumes A1 and A2 map to same cache block

Example

	P1			P2			Bus				Memory	
step	State	Addr	Value	State	Addr	Value	Action	Proc.	Addr	Value	Addr	Value
P1 Write 10 to A1	Excl.	A1	10				WrMs	P1	A1		A1	0
P1: Read A1	Excl.	A1	10								A1	0
P2: Read A1				Shar.	A1		RdMs	P2	A1		A1	0
	Shar.	A1	10				WrBk	P1	A1	10	A1	10
				Shar.	A1	10	RdDa	P2	A1	10	A1	10
P2: Write 20 to A1	Inv.			Excl.	A1	20	WrMs	P2	A1		A1	10
P2: Write 40 to A2							WrMs	P2	A2		A1	10
				Excl.	A2	40					A1	10

Assumes A1 and A2 map to same cache block,
but A1 != A2

Complications: Write Races

- **Cannot update cache until bus is obtained**
  - **Otherwise, another processor may get bus first, and then write the same cache block!**
- **Two step process:**
  - **Arbitrate for bus**
  - **Place miss on bus and complete operation**
- **If miss occurs to block while waiting for bus**
  - **handle miss (invalidate may be needed) and then restart.**
- **Split transaction bus:**
  - **Bus transaction is not atomic: can have multiple outstanding transactions for a block**
  - **Multiple misses can interleave, allowing two caches to grab block in the Exclusive state**
    - » **Must track and prevent multiple misses for one block**