Advanced Cache Coherency

Andy James

Joel McMonagle

Overview

- Scalable Coherency
  - Bus Based
  - Hierarchical
  - Directory Based
- Coherency State Machines
  - MSI and MEI
  - MESI and MOESI
  - KEIO
- Integrating Multiple Protocols
- Questions

Scalable Coherency

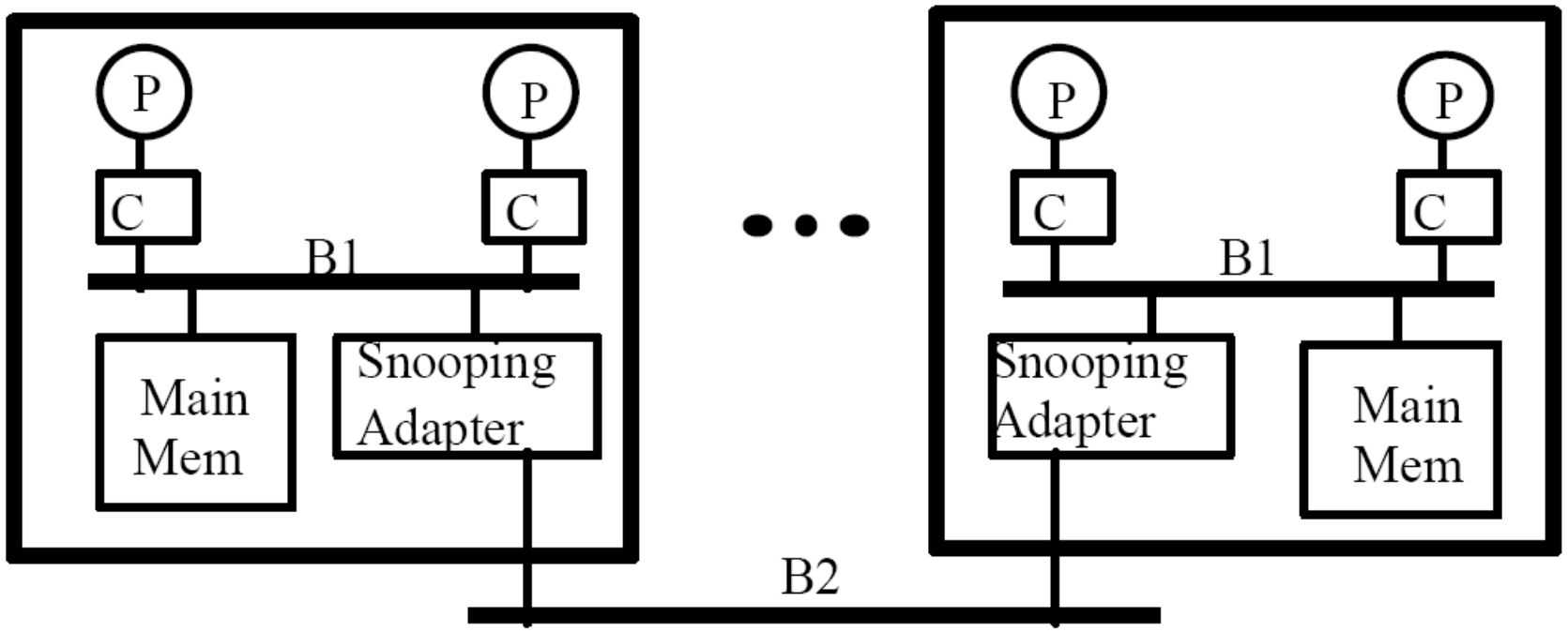
- Bus-Based
  - Scales poorly
- More Scalable Architectures
  - Hierarchical Snoopy Protocols
  - Directory Based Protocols

Bus Based Protocols

- Every processor is connected to a common bus
- Processors “snoop” on this bus to know what other processors are doing in memory
- Processors take action based on what they notice happening on the bus

Scalable Coherency:

Bus Based Protocols Example



Multiple CMPs (each with own internal on-chip bus) tied together by system bus.

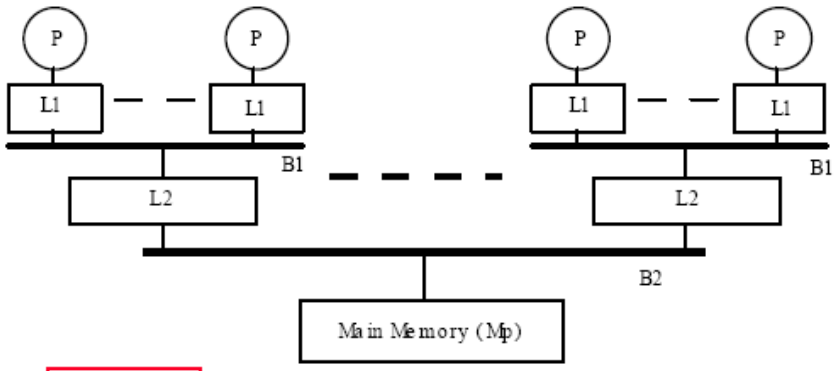
Scalable Coherency:

Hierarchical Protocols

- Extension of standard Bus Snooping
  - Utilizes a tree of buses or rings
  - Snooping message only propagates upward through the levels when necessary (based on snoop results for this level)
  - High latency
    - BW bottleneck at root of tree
    - Separate snoops at each level
  - Not popular due to low performance.

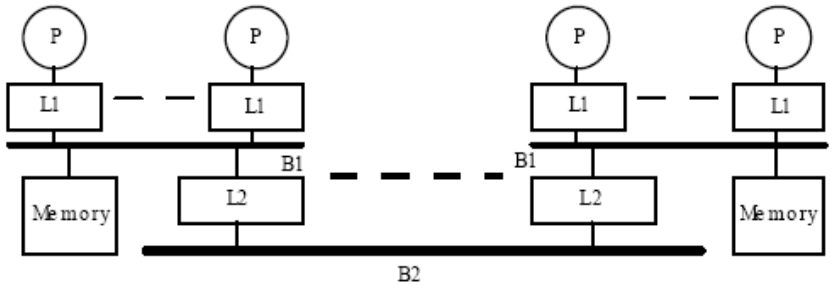
Scalable Coherency:

Hierarchical Protocols Example



UMA

(a)



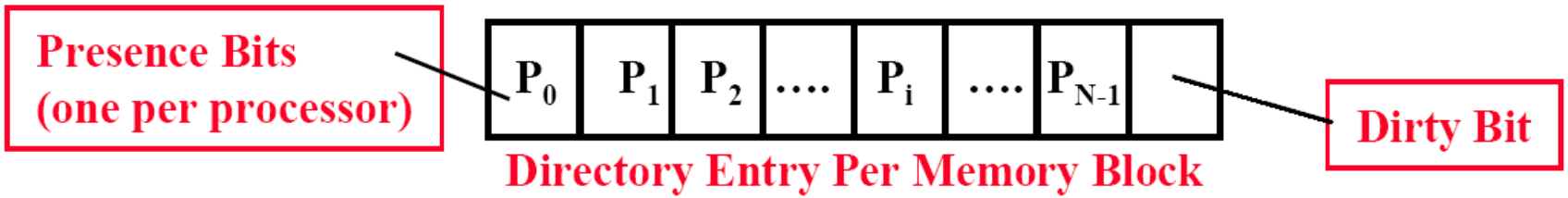
NUMA

(b)

Scalable Coherency:

Directory Based Protocols

- Each memory block has a directory entry
  - The entries have P+1 bits where P is the number of processors
  - Each processor is represented with a presence bit
  - There is one dirty bit per entry
    - If the dirty bit is on then only one presence bit can be on
- Nodes only communicate with other nodes that have the memory block



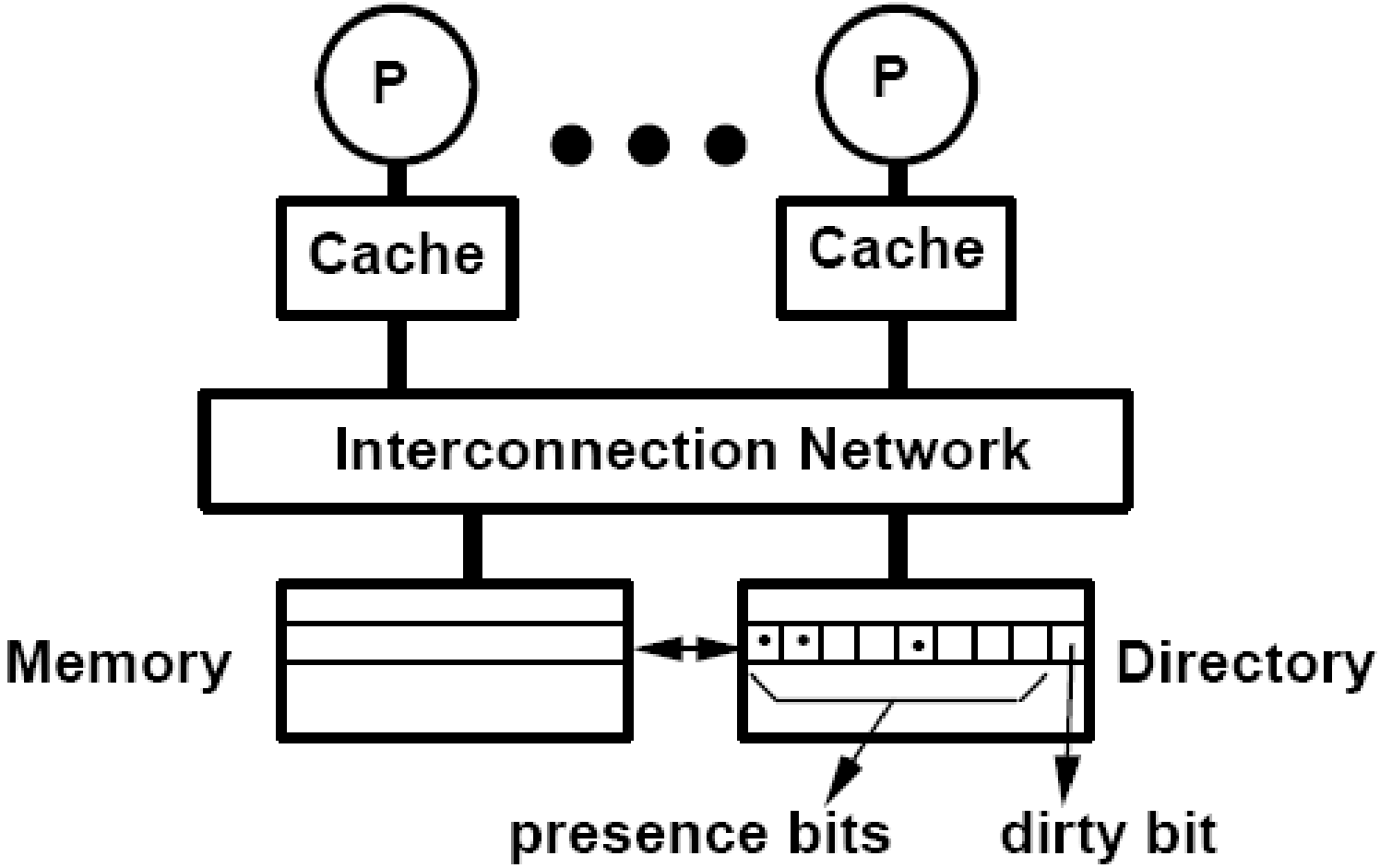
Scalable Coherency:

Centralized Directory Based Protocol

- The memory and directory are both centralized
- Poor scalability, but better than Bus Snooping

Scalable Coherency:

Centralized Directory Based Protocol Example



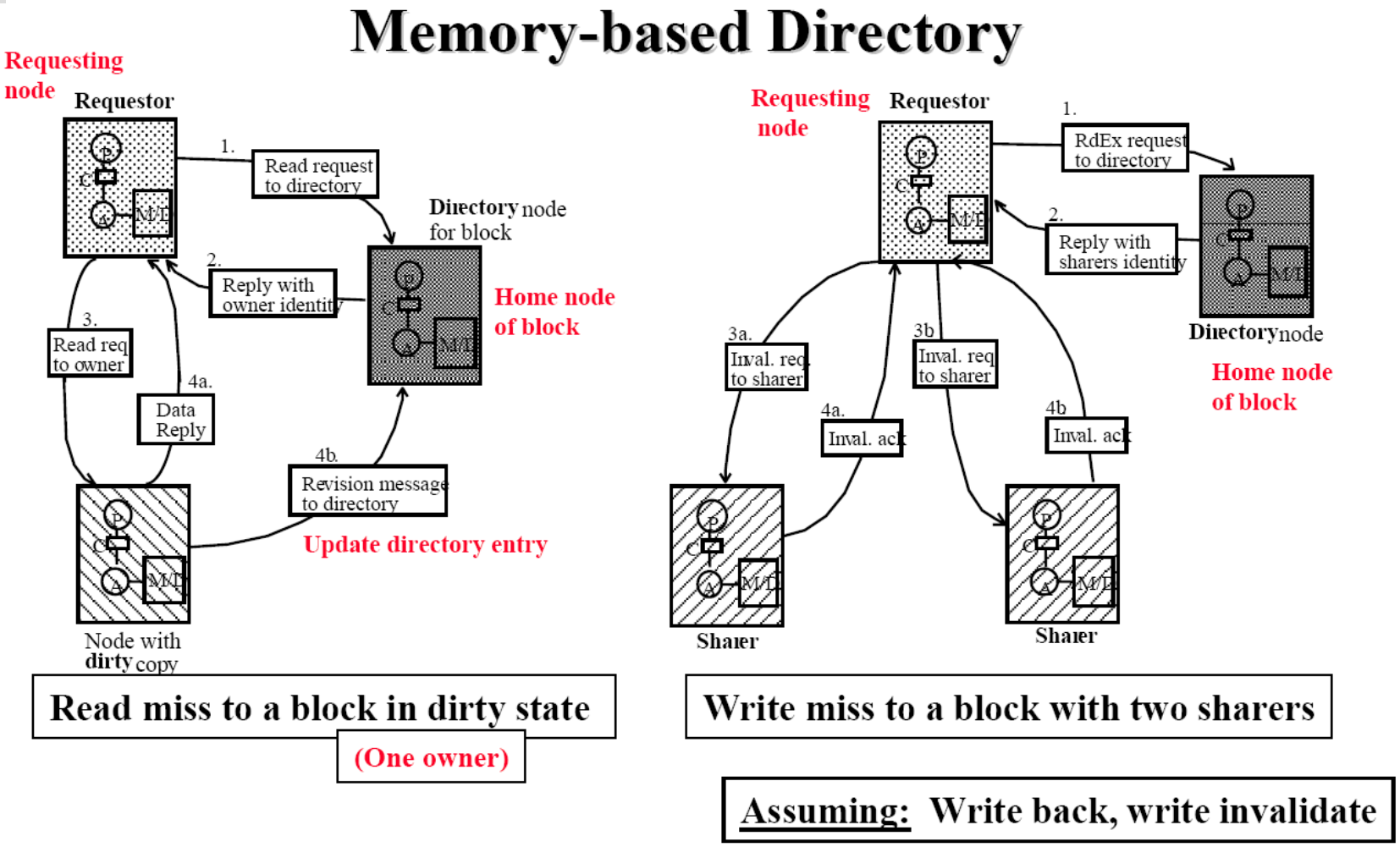
Scalable Coherency:

Decentralized Directory Based Protocols

- Each node stores small piece of entire directory corresponding to the memory resident at that node.
- Directory queries are directed to the node which the address of the block corresponds to.

Scalable Coherency:

Decentralized Directory Based Protocol Example



Coherency State Machines

- Nodes maintain their own state machines for each memory block.
- Most memory transactions involving the cache invoke the coherency protocol to send coherency messages to sharers of the corresponding block.
- Varying flavors of bus messages received via snooping trigger different transitions, based on scheme designed.

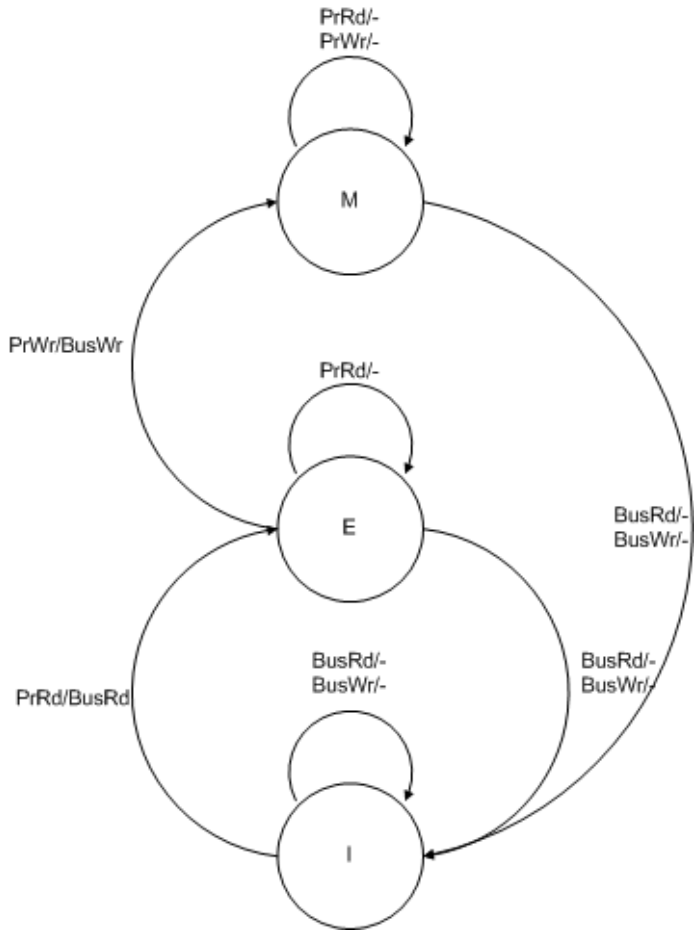
Coherency State Machines:

MEI

- Most basic implementation of a cache coherency state machine.
- Very restrictive – No sharing of blocks allowed at any time.
- Modified: You have modified data
- Exclusive: You have the only copy of the data
- Invalid: Your copy of the data is not up to date
- The valid and dirty bits of a cache line essentially implement MEI

Coherency State Machines:

MEI State Transition Diagram

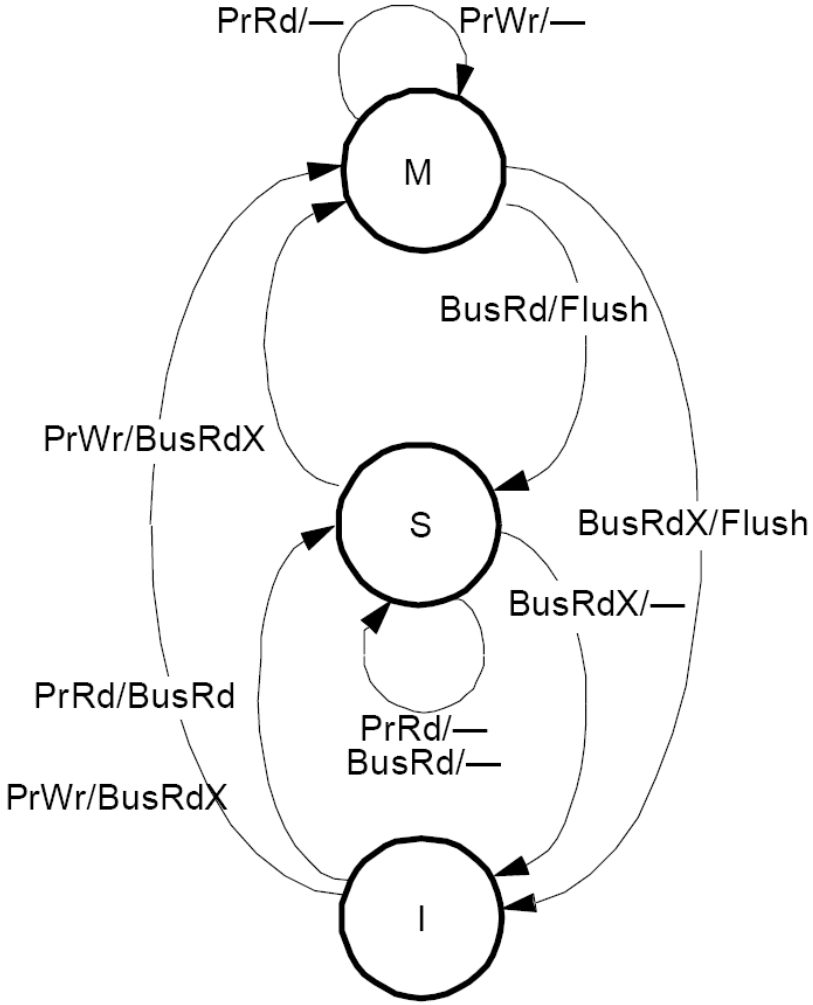


Review: MSI

- Very basic cache coherency protocol
  - But more robust than MEI – allows sharing.
- Modified: You have modified shared data
- Shared: You have a copy of data that another processor also has
- Invalid: Your copy of the data is not up to date

Coherency State Machines:

MSI State Transition Diagram

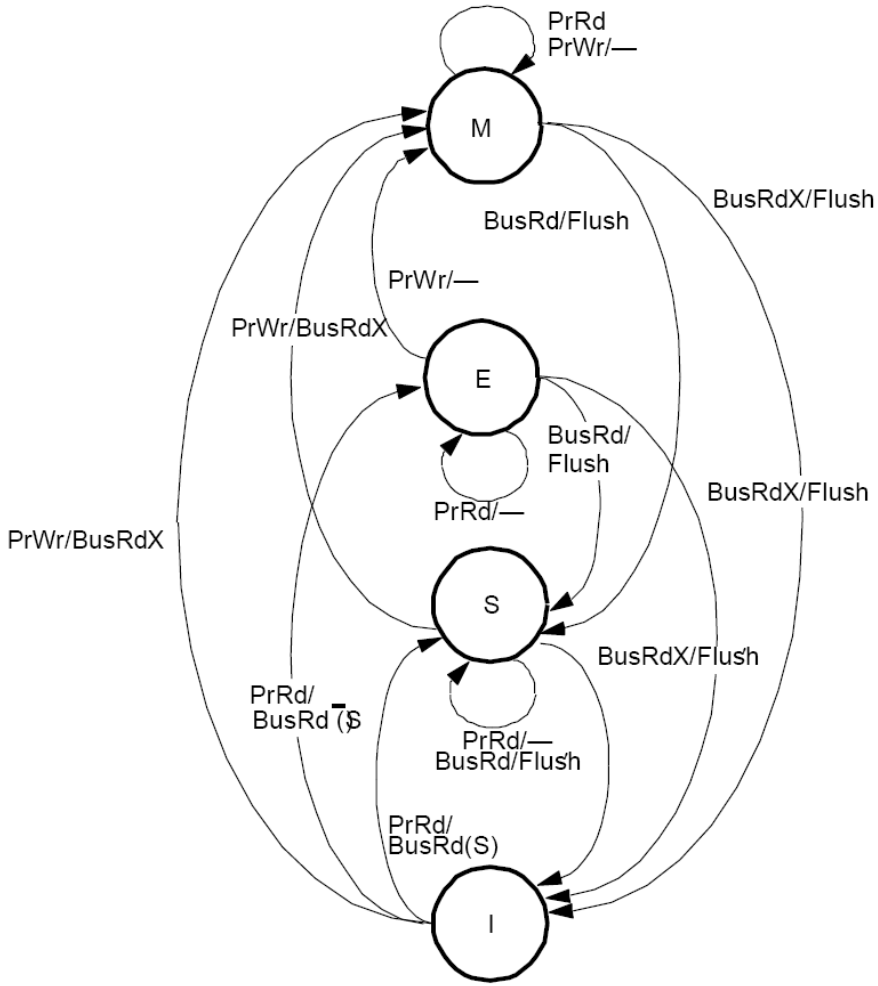


Review: MESI

- Extension of MSI.
  - Benefit: Reduces the number of bus messages sent out for I->M transition.
- Modified: You have modified shared data
- Exclusive: You are the sole owner of this data and are free to modify it without a bus message.
- Shared: You have a copy of data that another processor also has
- Invalid: Your copy of the data is not up to date

Coherency State Machines:

MESI State Transition Diagram



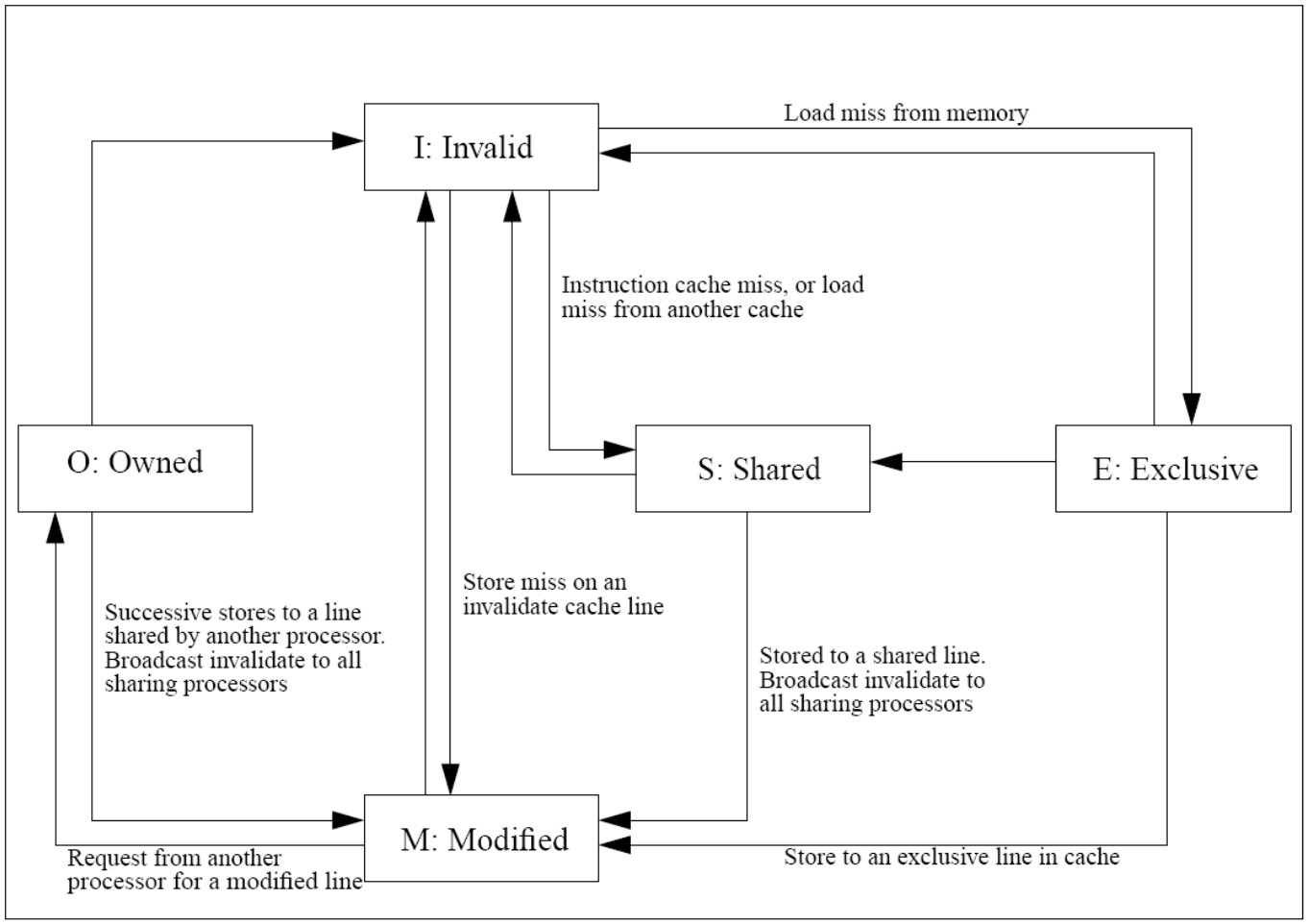
Coherency State Machines:

MOESI

- Further extension of MESI.
  - Benefit: Reduces the number of bus messages sent out for I->M transition while still allowing multiple sharers.
- Modified: You have modified shared data.
- Owner: Your data is shared, but you have the master copy in the cache, and can modify this data as you wish without a bus message.
- Exclusive: You are the sole owner of the data and are free to modify it also without a bus message.
- Shared: You have a copy of data that another processor also has.
- Invalid: Your copy of the data is not up to date.

Coherency State Machines:

MOESI State Transition Diagram



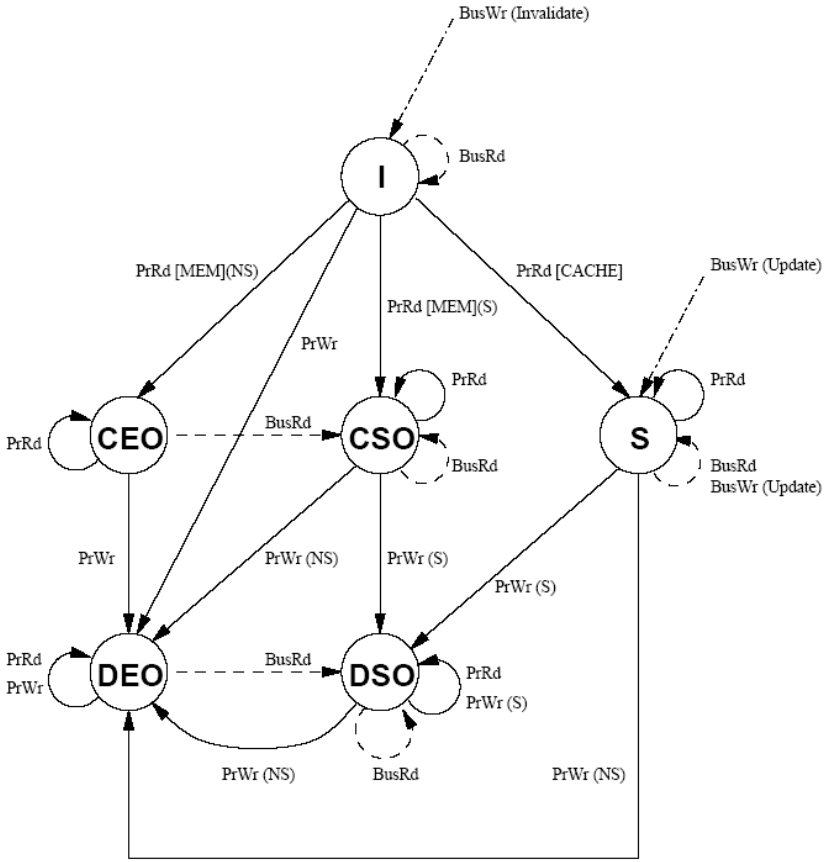
Coherency State Machines:

KEIO

- If no cache contains a particular line, memory owns it
- When a cache reads a line from memory it is the owner
- The owner of the line supplies the cache line to other caches
- Reading a line from another cache does not change ownership
- Writing a line owned by another cache does change ownership
- Dirty lines are written back to memory by the owner

when the cache line is flushed. Memory becomes the owner again

KEIO State Machine



Invalid (I)	Clean Exclusive Owner (CEO)	Dirty Exclusive Owner (DEO)
Shared (S)	Clean Shared Owner (CSO)	Dirty Shared Owner (DSO)

Industry Implementations

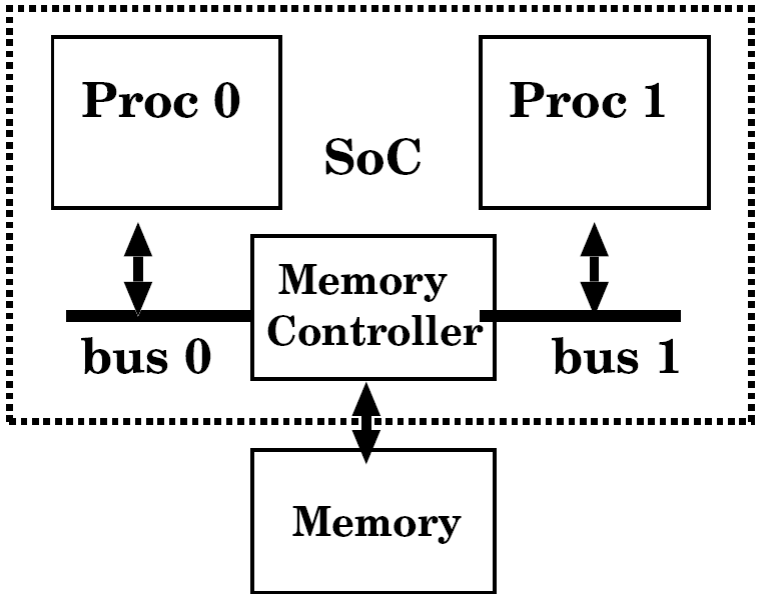
- Intel
  - MESI
- AMD
  - MOESI
- Sun Microsytems
  - JBus (MOESI)

Heterogeneous Systems and Cache Coherency

- In a heterogeneous system the different processors may be using different schemes natively
- How do we handle this?

Integration of Multiple Protocols

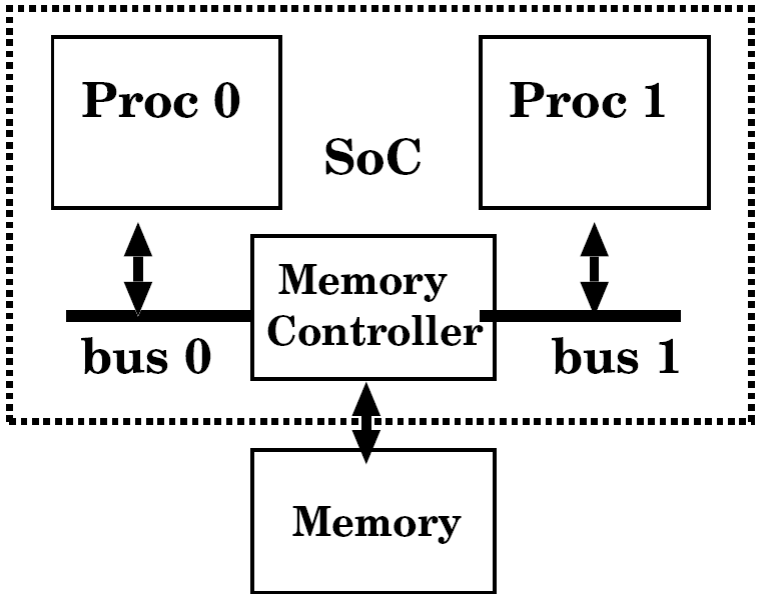
- Approach is simple: make the whole system seem as though it implemented the simplest of the protocols by modifying coherency messages.
- Example: Integration of multiple MPSoCs in a Multiprocessor Multiple Bus (MPMB) architecture.
- Memory controller maintains information about addresses in shared space
  - Address range
  - Apparent state of each processor for each block
  - Buffers cache blocks



Integration of Multiple Protocols:

MEI(P1) with MSI, MESI or MOESI (P0)

- Only 1 consideration needs to be made to prohibit (apparent) transition to S in P0
  - Convert P1 reads to appear as writes to P0
  - MC claims mastership of bus0 when request by P1 is within shared range and P0 is E for this block, according to MC.
  - MC drives a write on bus0 to force WB on P0 if dirty, or invalidation otherwise.
- Apparent protocol: MEI.

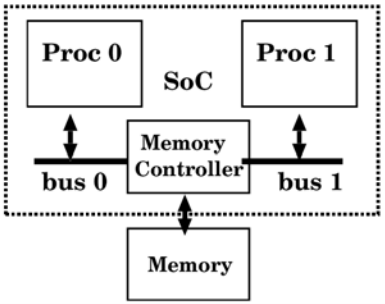


Integration of Multiple Protocols:

MSI(P1) with MESI or MOESI (P0)

- 2 considerations need to be made to prohibit E state in “apparent” protocol
  - P0 is forced to S instead of E by appropriate messages from MC.
  - O appears as S in MC
    - P1 in I state requests read, P0 in M state:
      P0 transitions to O locally and S apparently, and provides the block, which is buffered in MC and forwarded to P1 – memory NOT updated immediately.
      P1 transitions to S.

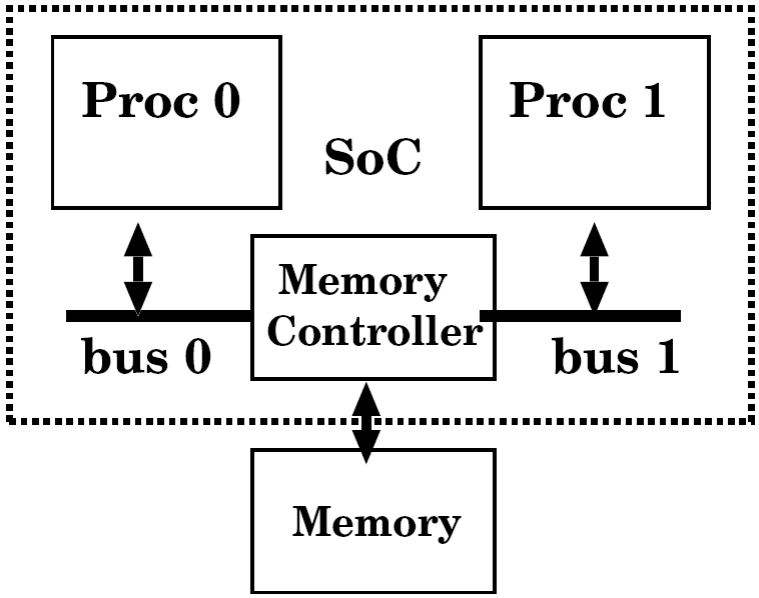
      P0 in I state requests read, P1 in M state:
      P1 transitions to S and provides the block, which is buffered in MC and forwarded to P1 and memory.
      P0 transitions to S.



- Apparent protocol: MSI.

MESI with MOESI

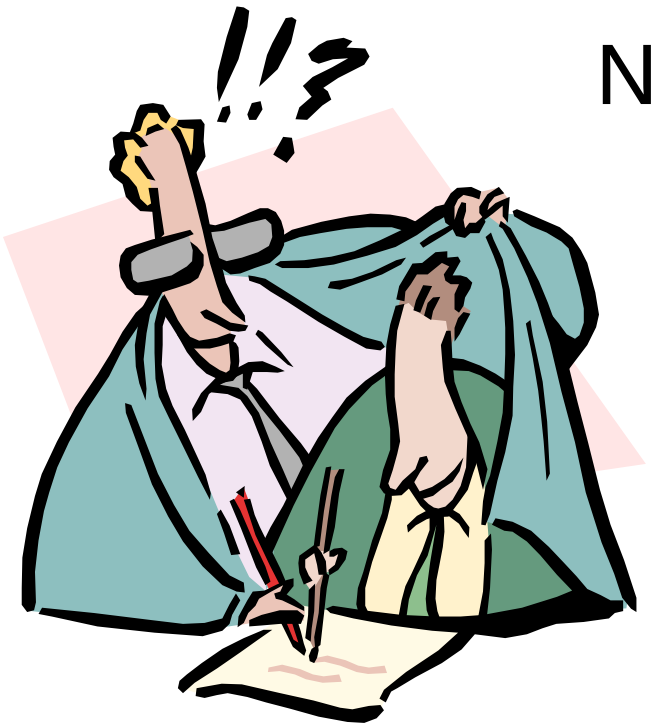
- Similar to the integration of MSI and MOESI, except E is not disallowed.
- Apparent protocol: MESI.



Integration with no native protocol

- A normal data cache without any native coherence protocol support behaves like it has the MEI protocol without any snooping capability.
  - At read miss, block is brought into the cache and valid bit set (E state).
  - Subsequent writes set the dirty bit (M state).
  - In WB cache, write misses set both the valid and dirty bits as cache entry is allocated (M state).
- Due to the lack of the snooping functionality, interrupts must be used to force WB or invalidation.

Questions?



No? Ok! ☺