CMPE110 Lecture 18
Cache Coherency

Heiner Litz

https://canvas.ucsc.edu/courses/19290

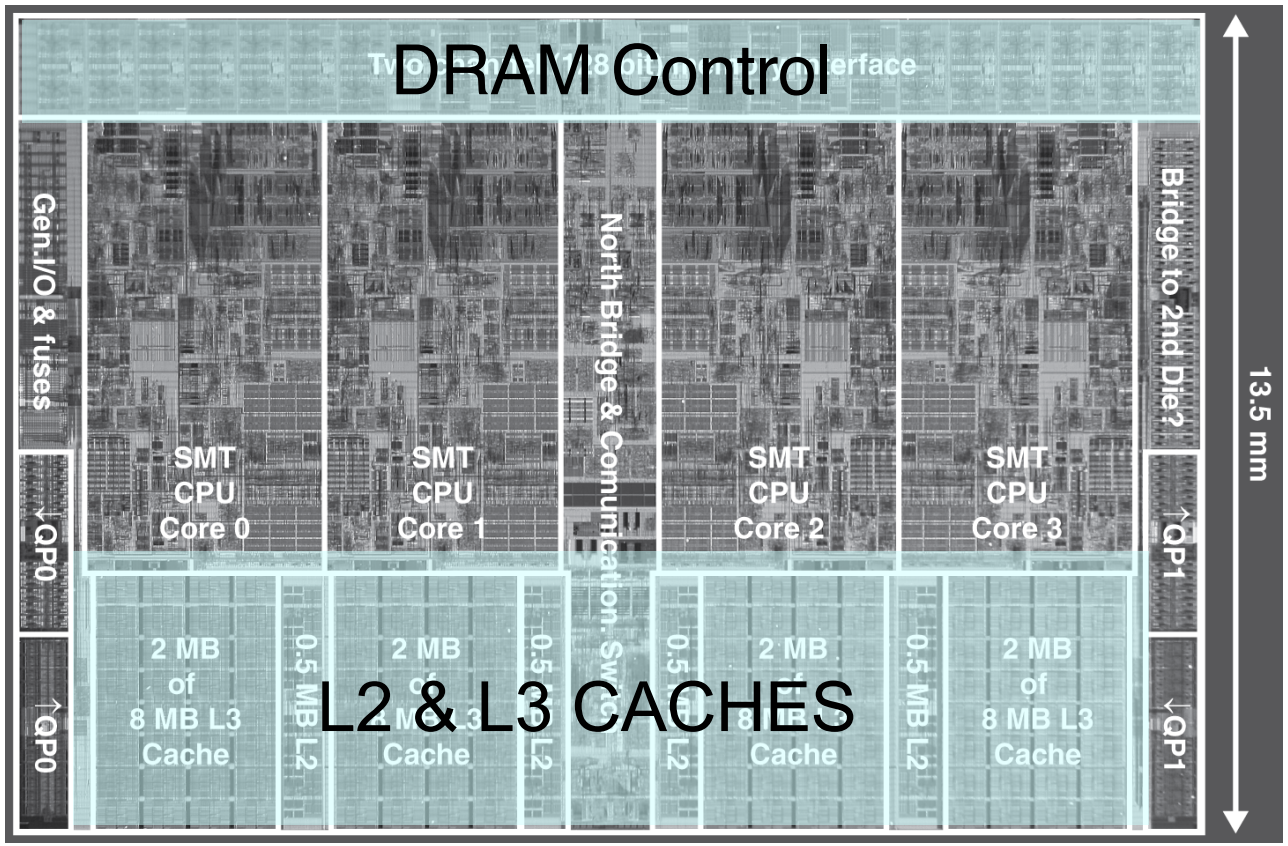
Announcements

- Readings: 5.1-5.4, 5.9-5.10

Review

Multilevel On-Chip Caches

Intel Nehalem 4-core processor

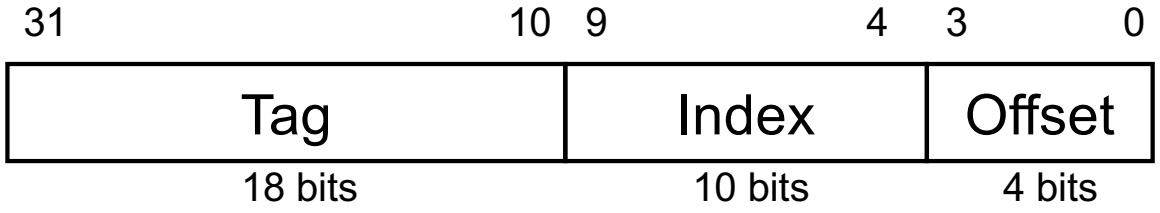


Per core:
-32KB, 4-way L1 $I
-32KB, 8-way L1 $D
-256KB, 8-way L2

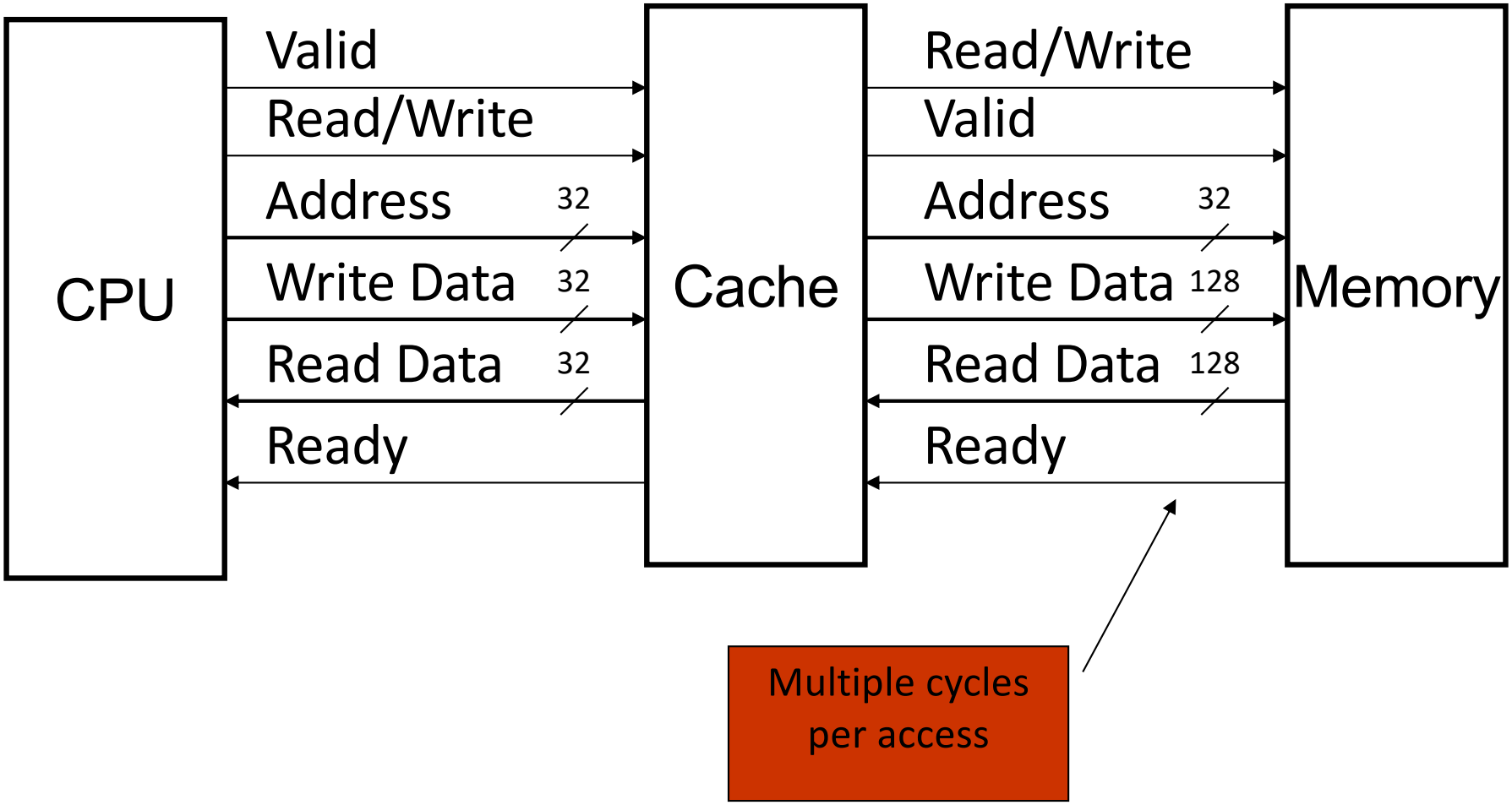
Shared
- 8 MB, 16-way L3

Cache Implementation: Control

- Example cache characteristics
  - Direct-mapped, write-back, write allocate
  - Block size: 4 words (16 bytes)
  - Cache size: 16 KB (1024 blocks)
  - 32-bit byte addresses
  - Valid bit and dirty bit per block
  - Blocking cache
    - CPU waits until access is complete

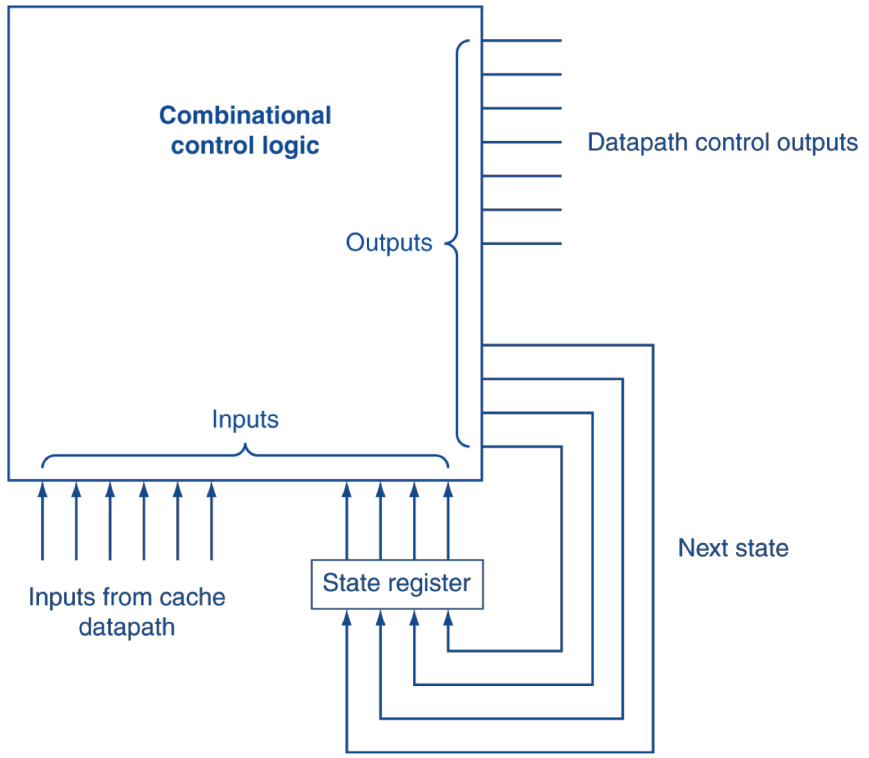


Interface Signals



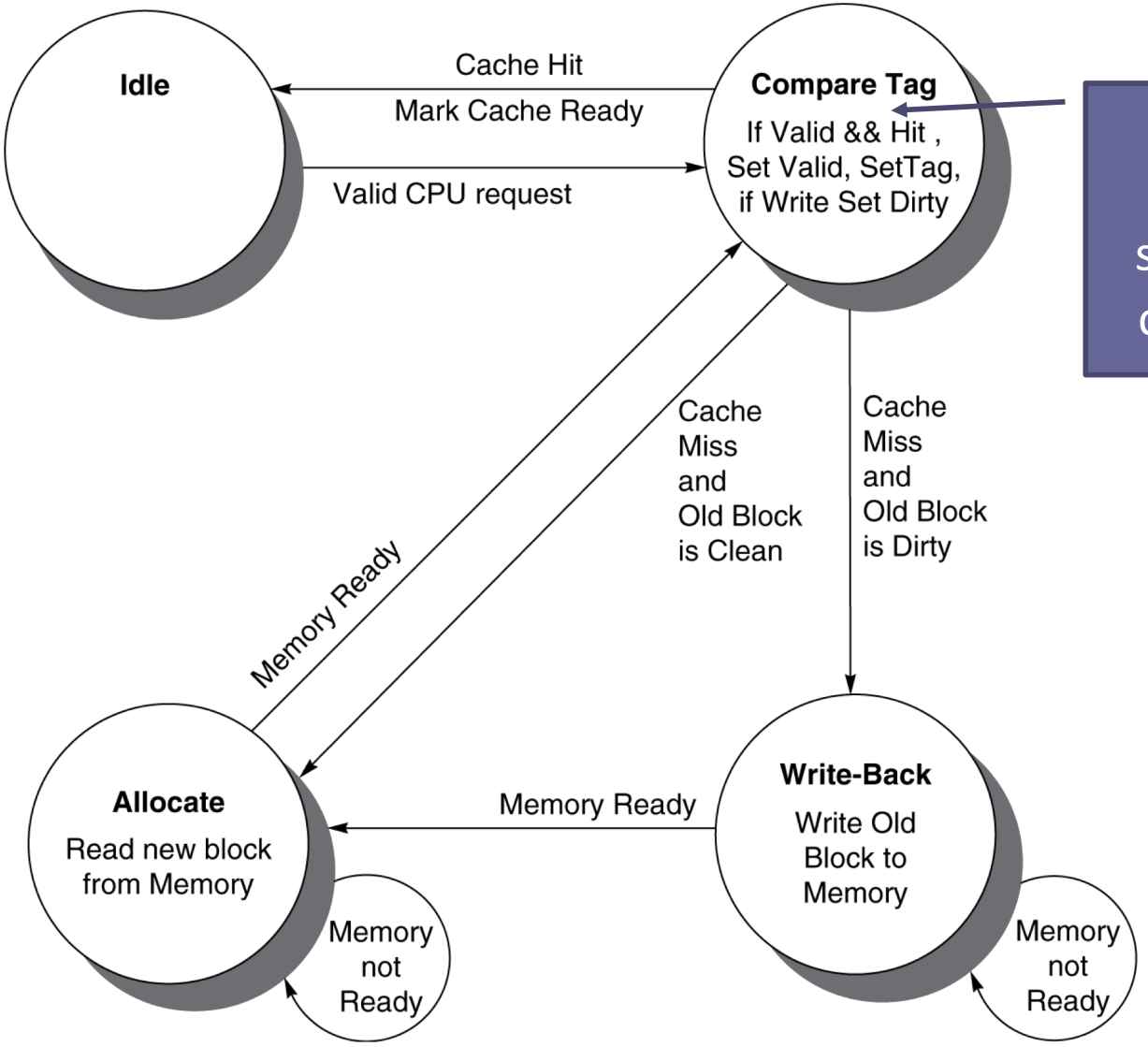
Reminder: Finite State Machines

- Use an FSM to sequence control steps
- Set of states, transition on each clock edge
  - State values are binary encoded
  - Current state stored in a register
  - Next state
    = fn (current state, current inputs)



- Control output signals = fn (current state)

Cache Controller FSM - WRITE

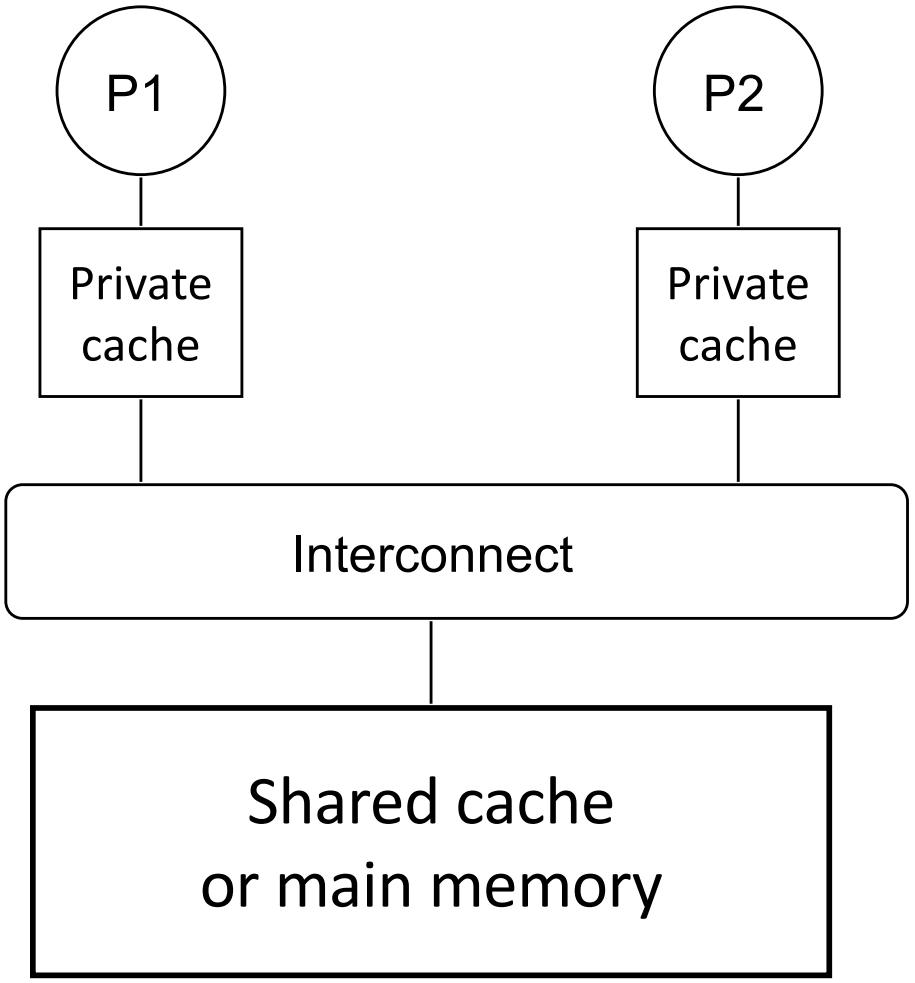


Could partition into separate states to reduce clock cycle time

Multi-core & Caches

- See any issues?

Cache Coherence Problem

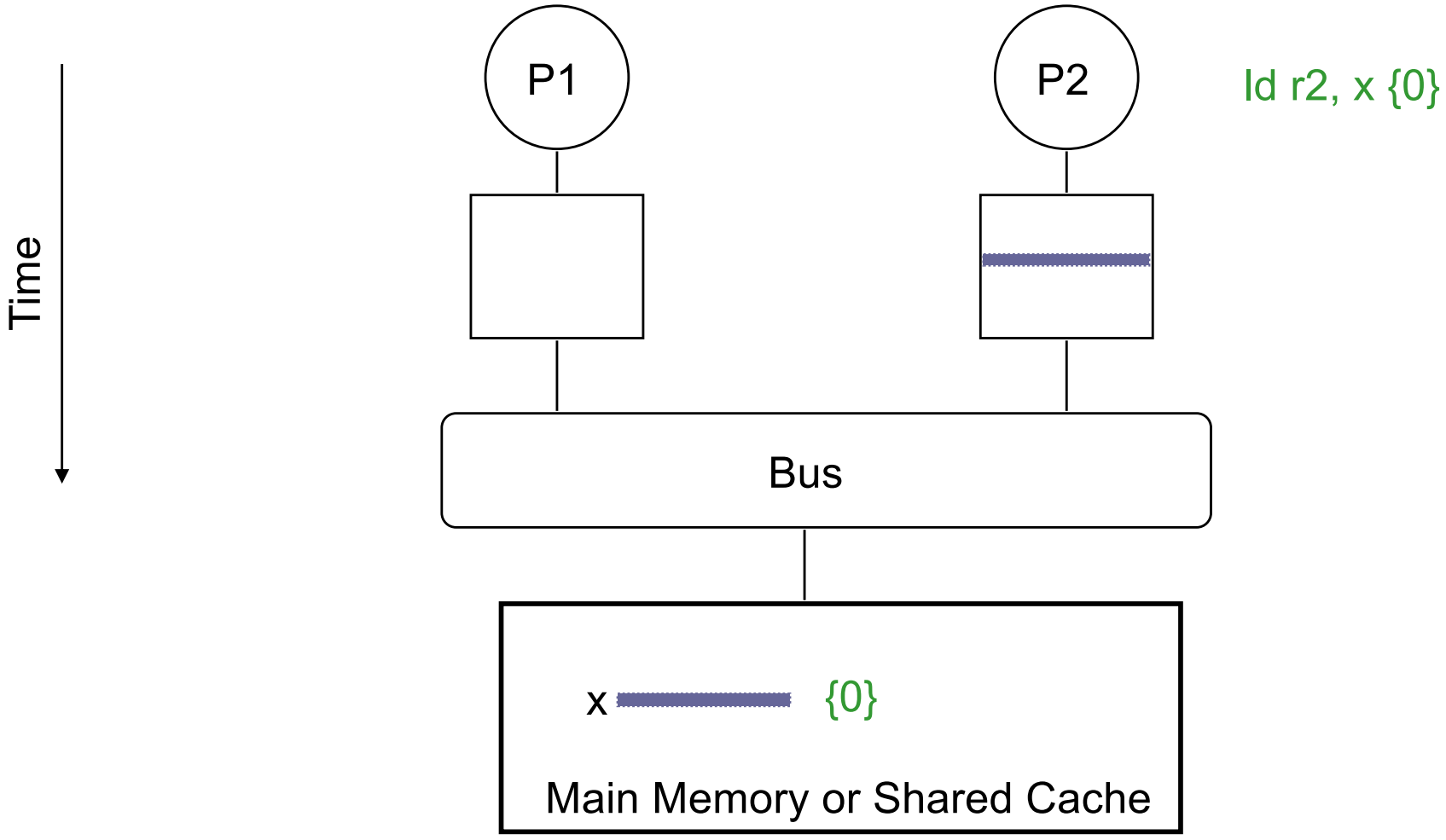


Cache Coherence

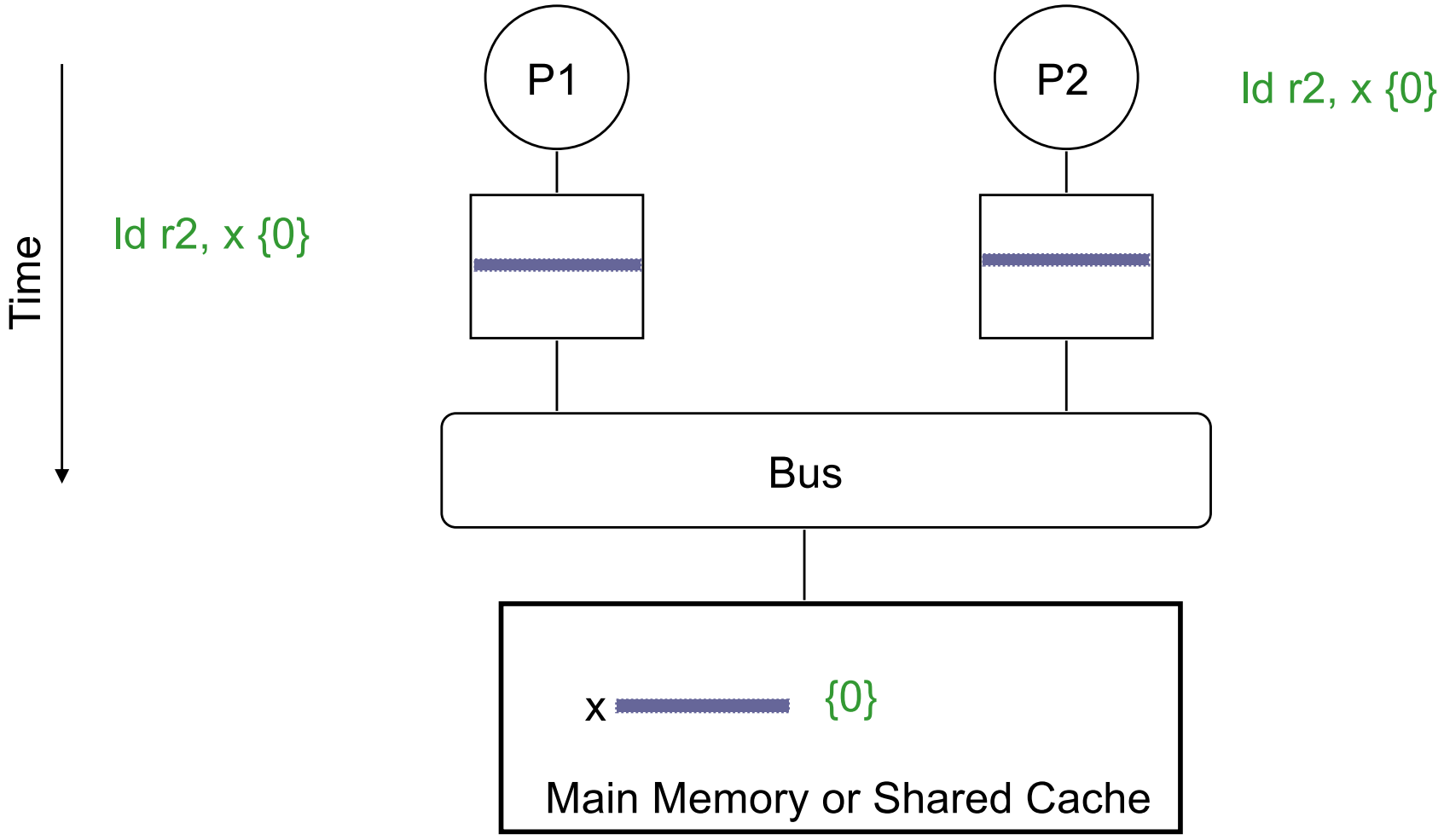
- Informally
  - Reads return most recently written value

- Formally
  1. P writes X; P reads X (no intervening writes)
     ⇒ read returns written value
  2. P1 writes X; P2 reads X (sufficiently later)
     ⇒ read returns written value
  3. P1 writes X, P2 writes X
     ⇒ all processors see writes in the same order
     ⇒ End up with the same final value for X

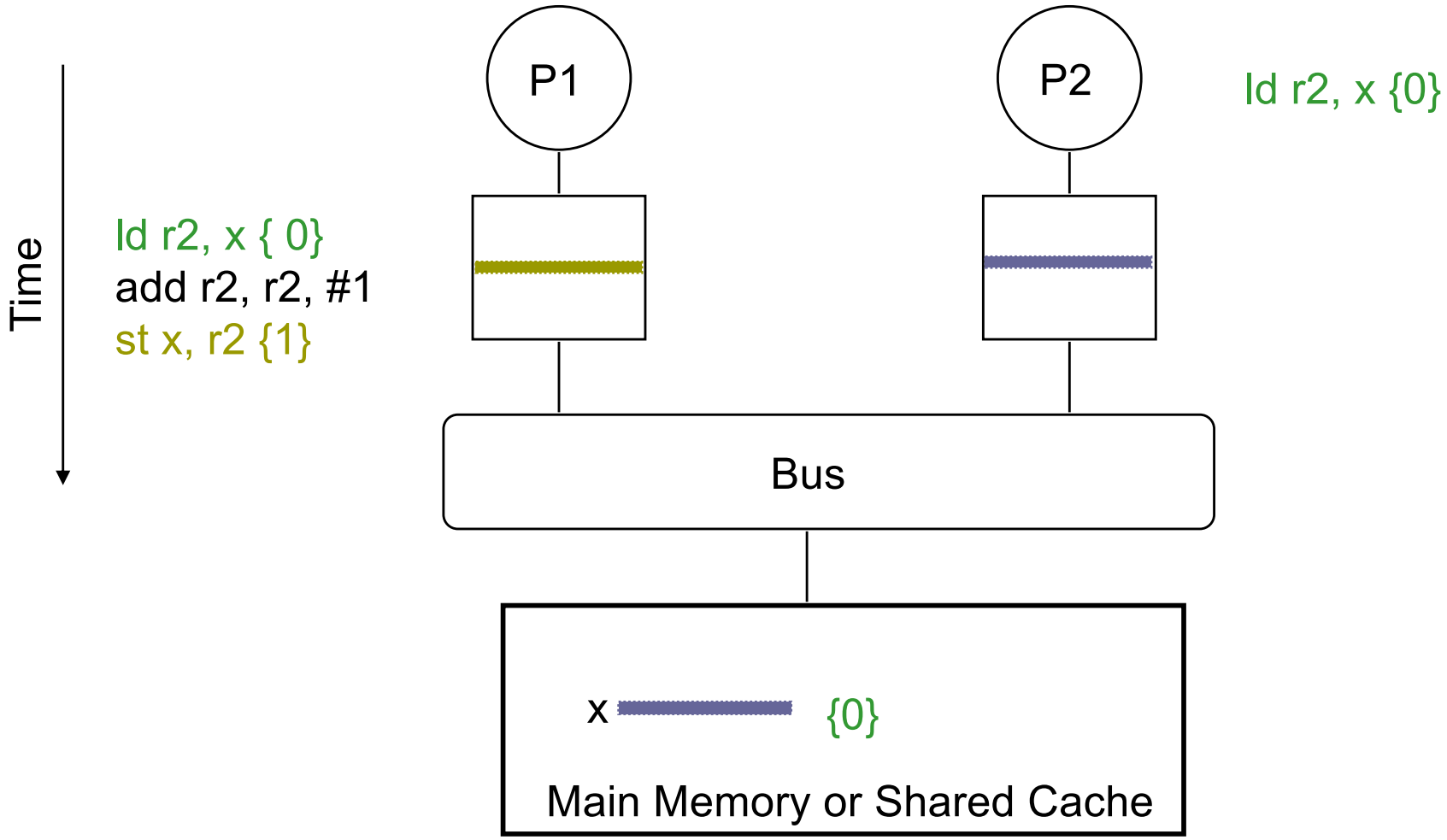
Cache Coherence Problem (Step 1)



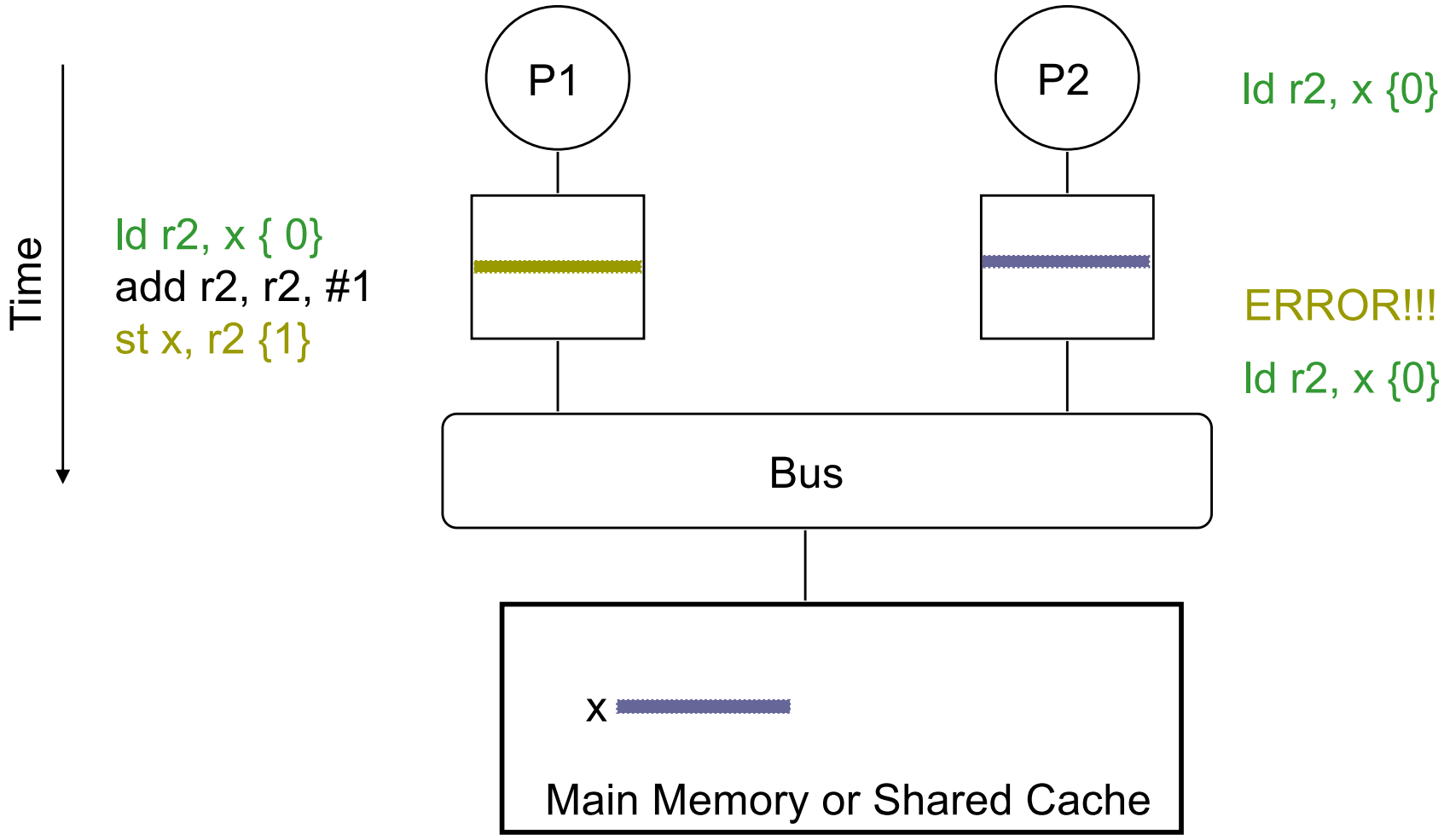
Cache Coherence Problem (Step 2)



Cache Coherence Problem (Step 3)



Cache Coherence Problem (Step 4)



Cache Coherence Protocols

- Cache coherence protocols
  - State and sequence of actions needed to ensure caches are coherence in multi-core systems
- Naïve protocol: turn off caching
  - Does not allow caching of read-shared data
  - Does not allow caching of private data
- Smarter protocol: allow one writer at the time
  - All caches can have copies of data while read-shared
  - On a write, invalidate all copies, allow a single writer

Cache Coherence Protocol

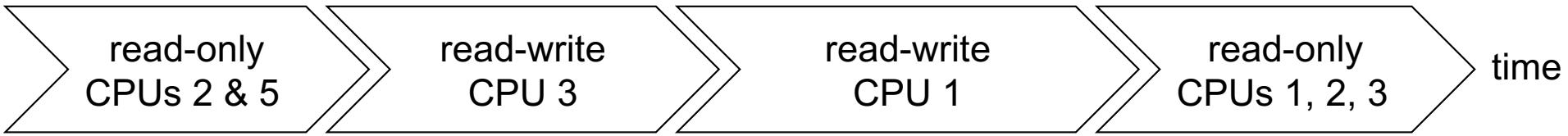
Divide lifetime of a memory value into epochs (time periods)

1. Single-Writer, Multiple-Read (SWMR) Invariant

   For any memory location A, at any given time (epoch), there exists only a single CPU that may write to A (and can also read it) or some number of CPUs that may only read A

2. Data-Value Invariant

   The value of the memory location at the start of an epoch is the same as the value of the memory location at the end of its last read–write epoch

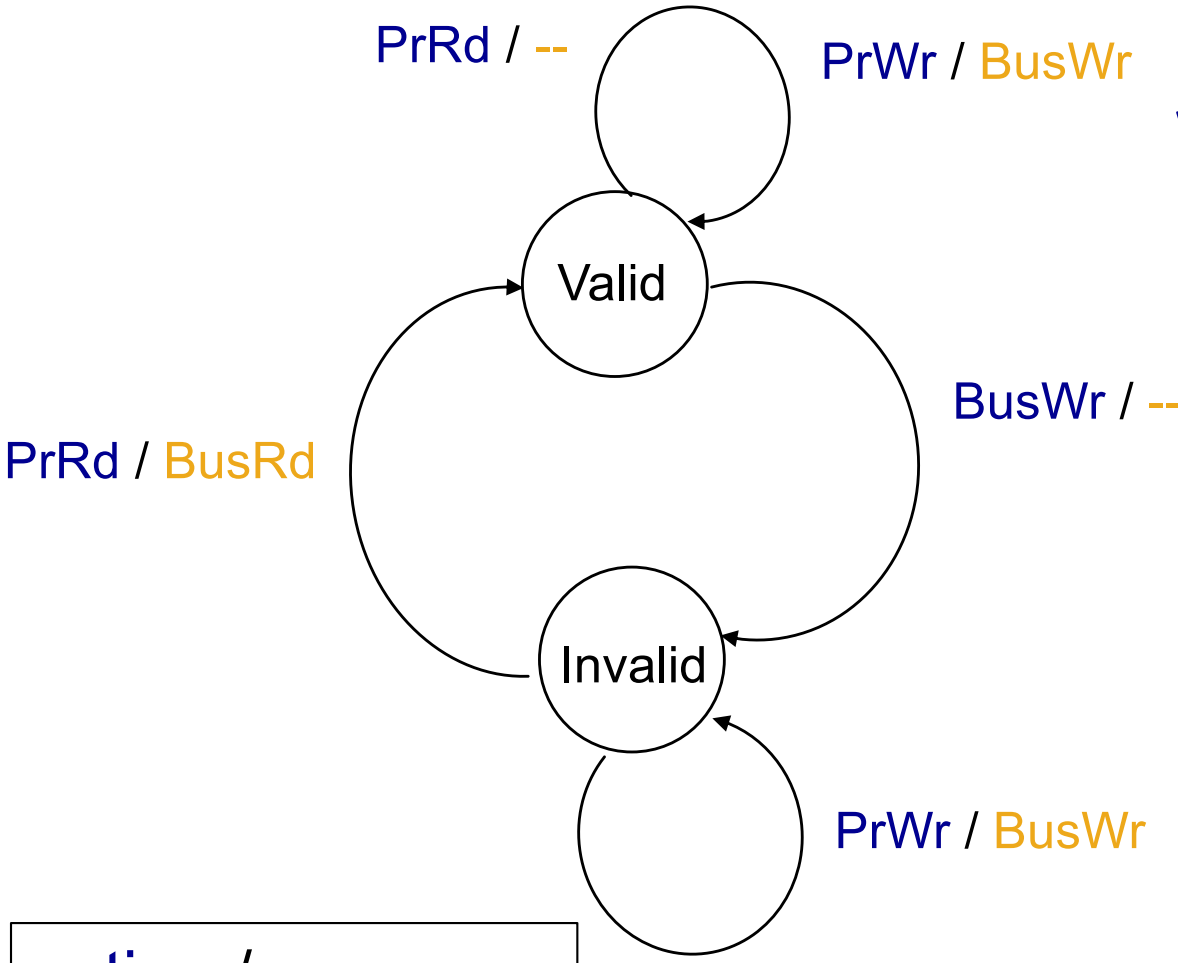


Implementation of Coherence Protocols

- Protocols rely on monitoring core actions
  - For now, assume all misses visible to all cores
    - Interconnect makes all actions visible
    - Cache FSM also reacts to requests from other caches
  - Known as a **snooping-based** protocols

- Protocols avoid stale copies
  - Whenever a core becomes a writer: invalidate copies

Simple Coherence Protocol



For write-through, no-write-allocate caches

Action	Abbreviation
Processor Read	PrRd
Processor Write	PrWr
Bus Read	BusRd
Bus Write	BusWr

action / response
(input / output)

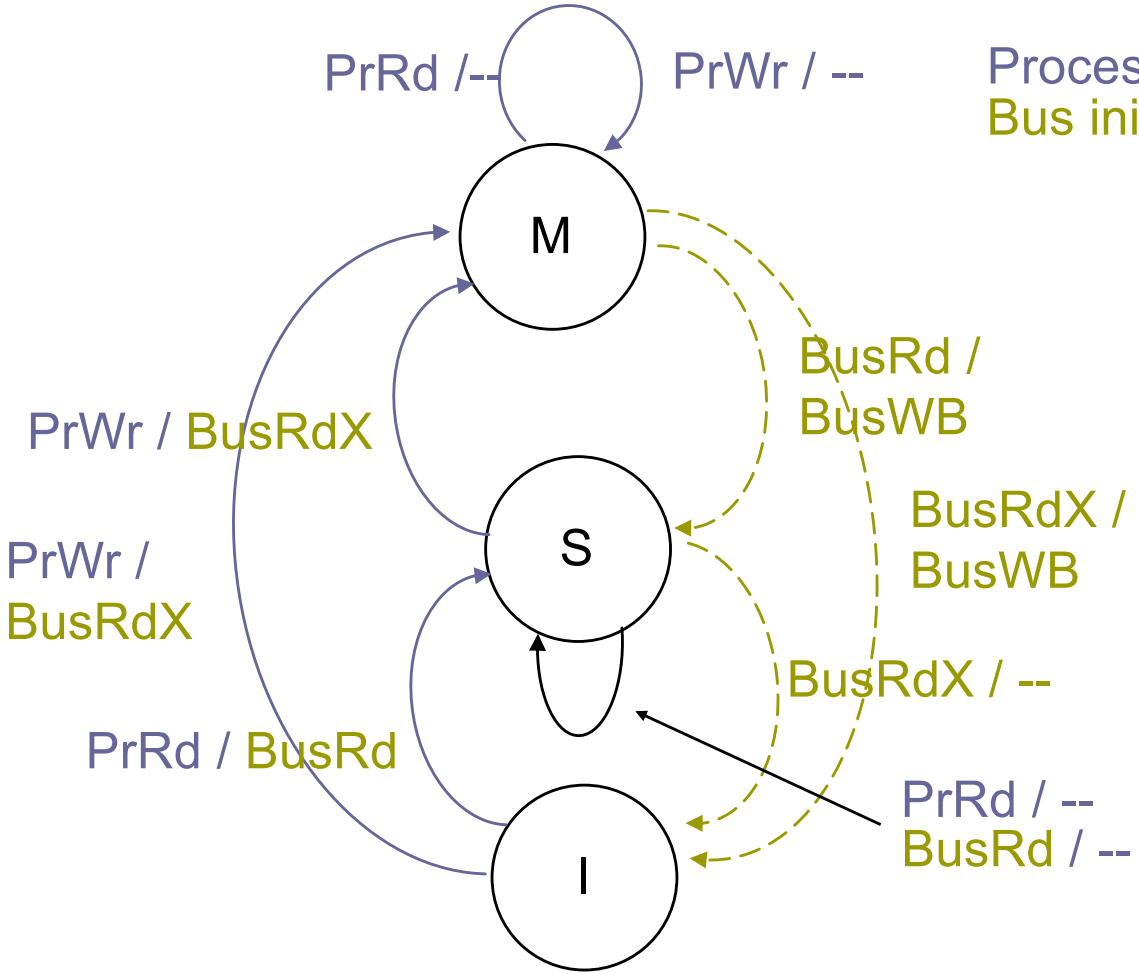
Is 2-state Protocol Coherent?

- Assume bus transactions and memory operations are atomic
  - Processor waits for memory operation to finish before issuing next
  - With one-level cache, assume invalidations applied during bus xaction
- SWMR Invariant
  - All writes go to bus + atomicity, causes transition to invalid state
- Data-Value Invariant
  - Read misses appear on bus, and will “see” last write in bus order
  - Read hits: do not appear on bus
  - But value read was placed in cache by either
    - Most recent write by this processor or most recent read miss
    - Both these transactions appeared on the bus
  - So reads hits also see values as produced bus order

A 3-State Write-Back Invalidation Protocol

- 2-State protocol
  - + Simple hardware and protocol (used in Sun Niagara)
  - - Bandwidth: write-through, every write goes on bus
    - Can work in multi-core with additional level of shared cache
- 3-State protocol (MSI)
  - Modified (called Exclusive in some books)
    - One cache has valid/latest copy
    - Memory is stale
  - Shared
    - One or more caches (and memory) have valid copy
  - Invalid
- Must invalidate all other copies before entering modified state
  - Requires bus transaction (order and invalidate)

MSI Coherence Protocol



Abbreviation	Action
PrRd	Processor Read
PrWr	Processor Write
BusRd	Bus Read
BusRdX	Bus Read Exclusive
BusWB	Bus Writeback