# Cache Coherence Protocols: Evaluation Using a Microprocessor Simulation Model

James Archibald and Jean-Loup Baer
CS258 (Prof. John Kubiatowicz)
March 19, 2008

Presentation by: Marghoob Mohiuddin

# Outline

- Cache coherence protocols for shared bus multiprocessors
  - Write-back caches
  - Write-once, Synapse, Berkeley, Illinois, Firefly, Dragon
- Simulation
  - Workload modeled probabilistically
    - Private blocks and shared blocks
    - Cache hits, misses occur with fixed probability

# Write-Once

- Dirty → mem write on replace
  - Reserved is dirty, but up to date in memory
- *Invalidates*
- Read miss:
  - Dirty copy or from memory
  - Dirty → Valid
- Write hit:
  - *No bus transaction if written once* (Reserved → Dirty, Dirty → Dirty)
  - Valid → mem write, other caches invalidate
- Write Miss:
  - Dirty copy or from memory
  - Other caches invalidate

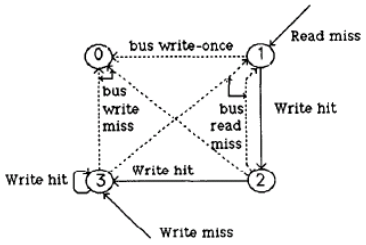



Fig. 1. Write-once transition diagram.

# Synapse

- Dirty → mem write on replace
- *No invalidates*
- Owner:
  - Cache with Dirty copy or memory
- 1-bit tag per block in memory
  - Memory owns the block
- Block always comes from memory
- Read miss:
  - Dirty copy written to memory
  - Dirty → Invalid
- Write hit:
  - Dirty → no bus transaction
  - Valid → treat as write miss
- Write Miss:
  - Same as read miss
  - Load as Dirty

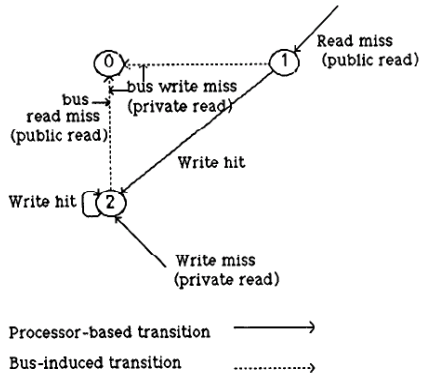



Fig. 2. Synapse transition diagram.

# Berkeley

- Dirty/Shared-Dirty ➔ mem write on replace
- *Invalidations, cache-to-cache transfers*
- Dirty blocks not written to memory on being shared
- Read miss:
  - Owner supplies block
  - Dirty → Shared-Dirty
- Write hit:
  - Invalidate other copies
  - Change to Dirty
- Write miss:
  - Owner supplies block
  - Invalidate other copies
  - Change to Dirty

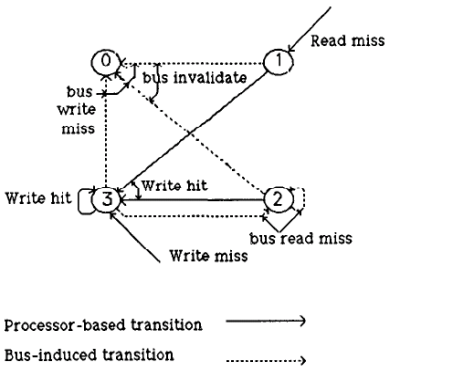



Fig. 3. Berkeley transition diagram.

# Illinois

- Dirty ➔ mem write on replace
- *Invalidations, requesting cache able to determine block source*
- Read miss:
  - Cached copy if possible
    - Dirty copy written to memory
    - All copies now Shared
  - No cached copies ➔ Valid-Exclusive
- Write hit:
  - Shared copies invalidated
- Write miss:
  - Similar to read miss
  - Other copies invalidated

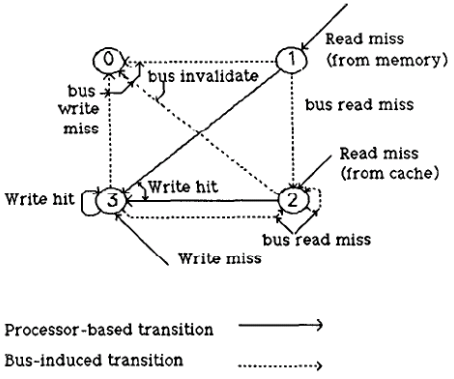



Fig. 4. Illinois transition diagram.

# Firefly

- Dirty ➔ mem write on replace
- *No invalidations, SharedLine*
- Read miss:
  - Cached copy supplied if possible
    - SharedLine raised
    - Dirty block written to memory
  - No cached copies ➔ Valid-Exclusive
- Write hit:
  - *Shared ➔ Write to memory*
    - Shared copies updated
    - SharedLine decides Valid/Valid-Exclusive
- Write Miss:
  - Cached copy if possible
    - Write on bus to update shared copies

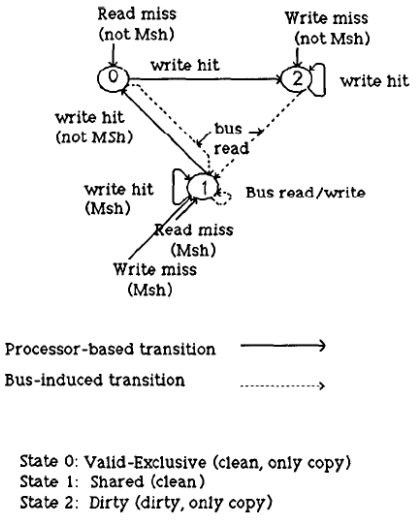



Fig. 5. Firefly transition diagram.

# Dragon

- Shared-Dirty/Dirty ➔ mem write on replace
- *No invalidations, SharedLine*
- Read miss:
  - Dirty copy or from memory
    - SharedLine decides Shared-Clean/Valid-Exclusive
- Write hit:
  - No mem write
  - Shared ➔ caches update copy
    - SharedLine decides Shared-Dirty/Dirty
- Write miss:
  - Cached copy if possible
    - Write bus to update shared copies

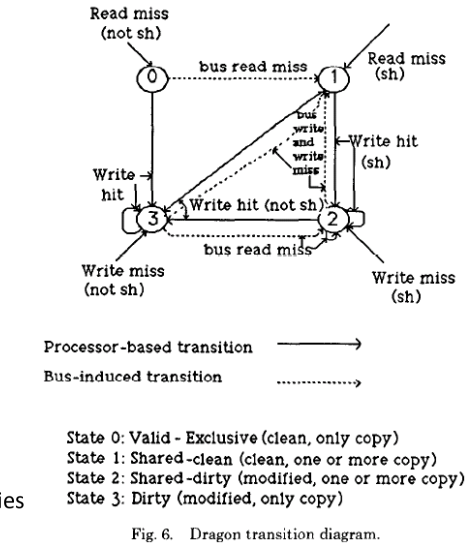



Fig. 6. Dragon transition diagram.

# Simulation Model: Multiprocessor

- Processor:
  - Work for *w* cycles, generate mem request, wait for response from cache
- Cache:
  - Bus commands higher priority than processor requests
- Bus:
  - Service requests from caches in FIFO order
  - Requests:
    - read miss, write miss, dirty block write back, request-for-write permission/invalidate/write broadcast

# Simulation Model: Workload

- Shared and private cache blocks
  - Private never present in other caches
- Processor generates reqs:
  - P(shared)=*shd*, P(read)=*rd*
- Private block reqs modeled probabilistically
  - P(hit)=*h*, write hit ➔ P(modified)=*wmd*
- Fixed num of shared blocks represented explicitly
  - Higher prob. of accessing a recently accessed block
  - More blocks ➔ less actual sharing
- Replacement
  - P(shared block chosen)$\propto$ no. of shared blocks in cache
  - P(private block replaced modified)=*md*
  - Blocks chosen at random
- *md, wmd, rd* not independent

# Simulation

- Memory/cache mismatch small compared to today
- Small caches
- Cache stalls until full block loaded
  - Block = 4 words
- Invalidate takes 1 cycle
- Run for 25000 cycles
- System power
  - Sum of proc. Utilizations
- Write-through also simulated
  - No write-allocate

Table I. Summary of Parameters and Ranges

| Parameter | Range |
|---|---|
| *shd* | 0.1–5% |
| *rd* | 70–85% |
| *h* | 95–98% |
| 1-*wmd* | 1.75–5.26% |
| *md* | 30–40% |
| *w* | Uniform [0..5] ~ |
| Main memory cycle time | Four cache cycles |
| Block size | Four words |
| Cache size | 2–16 kbytes |
| Number of shared blocks | 16–1024 |
| Number of processors | 1–15 |

# Simulation Results: Private Block Handling

- Efficiency in handling private blocks
  - Write hits to unmodified blocks
    - Illinois, Firefly, Dragon efficient due to Valid-Exclusive state
    - Berkeley has 1 cycle invalidate overhead
    - Write-once has mem write overhead for 1 word
    - Synapse has mem write overhead for 1 block
    - Write-once, Synapse have high overhead if memory latency is 100s of cycles
  - Replacement strategy
    - Write-once: P(mem write for repl. block) smaller
      - Written-once blocks up to date in memory

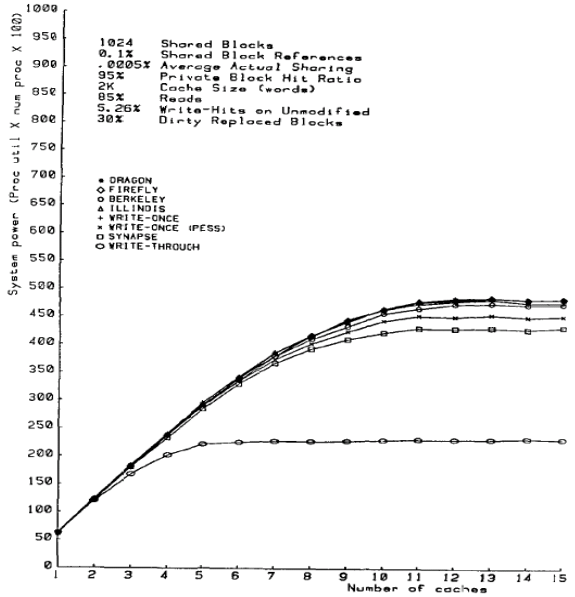
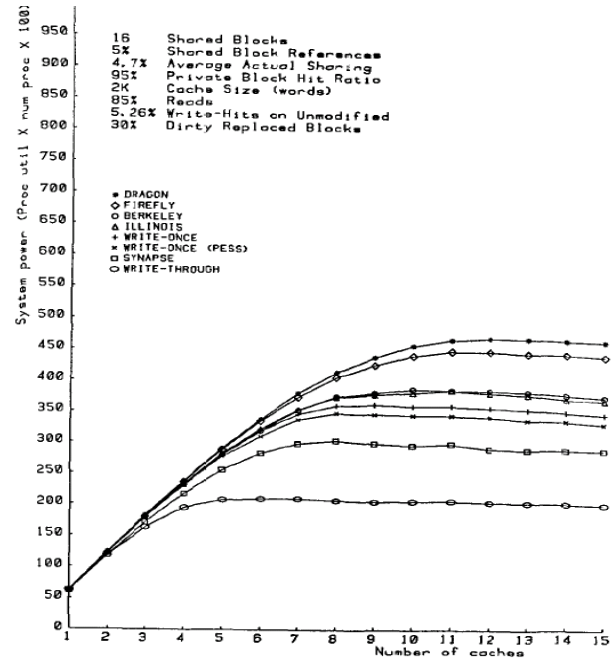



Figure 7

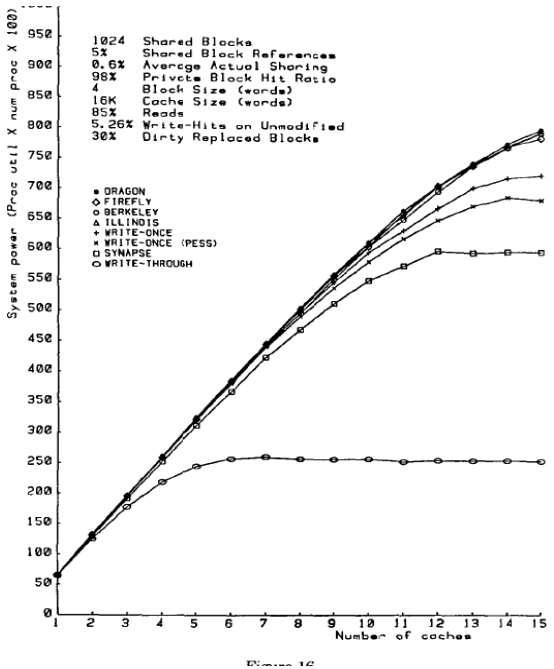
**Simulation Results: Private Block Handling**

# Simulation Results: Shared Block Handling

- Efficiency in handling shared blocks
  - Dragon and Firefly best
    - Updates instead of invalidates
    - Performance decreases with decreasing contention
      - Cache hit rates decrease due to increased no. of shared blocks
    - Firefly has overhead of mem write on write hit
- Berkeley beats Illinois (under high contention)
  - Illinois updates main memory on a miss for a dirty block
- Write-once low performance
  - Memory update on a miss for dirty block




**Simulation Results: Shared Block Handling**




**Simulation Results: Shared Block Handling**