ELE 655
Microprocessor System Design

Section 4 – Thread Level Parallelism

Class 1 – Intro
Cache Coherence

TLP
Thread Level Parallelism

- Background
  - Limitations on Clock frequency
  - Need for more performance
  - Increasing independent processing requests
  - Shift to the Cloud
  - Replication of HW vs Innovation

  → Thread Level Parallelism

TLP
Thread Level Parallelism

- Background
  - Thread Level Parallelism
    - Defined at a higher level than DLP
    - Controlled by OS with a single memory space
    - Multiple processes, each with multiple threads
    - Separate threads → separate program counters
    - Inter process communication
      - Shared data
    - Instead of context switching on a single processor

      → MIMD

TLP
Thread Level Parallelism

- Background
  - Software models
    - Multiple threads to achieve a single task

      parallel processing

    - Multiple threads to achieve different tasks
      - Potentially from different users
      - Single application split across processors
      - Multiple applications running independently

      request level parallelism

      multiprogramming

TLP
Thread Level Parallelism

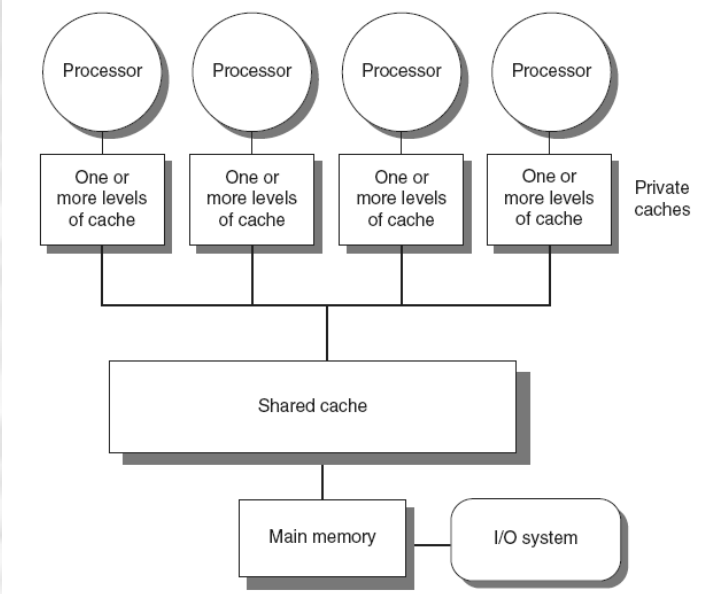
- MIMD
  - Multiple processors on a single chip
    Multicore
  - Multiple processors on separate chips
    - Some of which may be multi-core
      Multiprocessor
  - Multi-threading on a single core (SMT)

TLP
Thread Level Parallelism

- Multi-processor Architecture
  - Need n threads to take full advantage of n processors
  - Amount of computation assigned to a thread – grain size
    - TLP – 100K+ instructions / thread
  - Shared memory

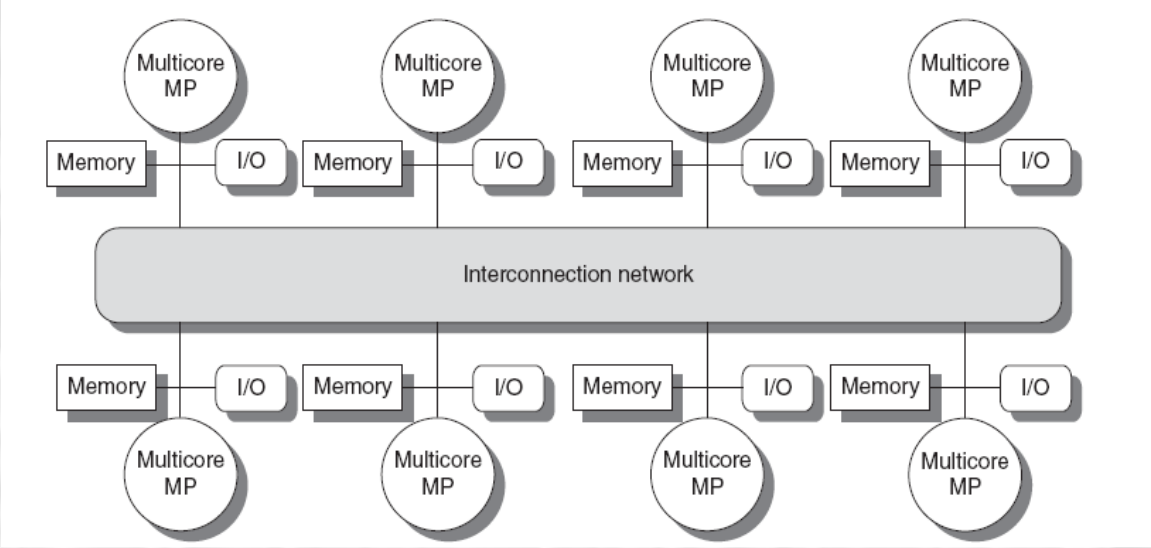
TLP
Thread Level Parallelism

- Multi-processor Architecture
  - Symmetric Multiprocessor
    - Relatively fewer processors
    - Centralized shared memory
    - Unified memory access (UMA)



TLP
Thread Level Parallelism

- Multi-processor Architecture
  - Distributed Shared Memory Multiprocessor
    - Relatively larger numbers of processors
    - Distributed memory (NUMA)
    - Interconnect network



TLP
Thread Level Parallelism

- Parallel Processing
  - Metrics
  - Example pg 349

TLP
Thread Level Parallelism

- Parallel Processing
  - Metrics
  - Example pg 350

TLP
Thread Level Parallelism

- Parallel Processing
  - Primary bottlenecks to performance
    - Memory Access
    - Access to shared information (memory)

TLP
Thread Level Parallelism

- SMP and Centralized Memory
  - Private Data
    - Used only with one processor
    - Cached locally
  - Shared Data
    - Used by multiple processors
    - Shared in memory
    - Cached locally to speed up access
    - But may be replicated in multiple locations
    - → Coherency issues

TLP
Thread Level Parallelism

- Cache Coherency
  - Global State
    - The value of a memory location held in main memory
  - Local State
    - The value of a memory location held in a local memory
    - May be multiple local values – 1 in each local memory
  - L1, L2 may be local (private) – L3 may be global (shared)

TLP
Thread Level Parallelism

- Cache Coherency

Time	Event	Cache contents for processor A	Cache contents for processor B	Memory contents for location X
0				1
1	Processor A reads X	1		1
2	Processor B reads X	1	1	1
3	Processor A stores 0 into X	0	1	0

TLP
Thread Level Parallelism

- Cache Coherency
  - Coherence
    - All reads by any processor must return the most recently written value
    - Writes to the same location by any two processors are seen in the same order by all processors (write serialization)
  - Consistency
    - When a written value will be returned by a read (delay)
    - If a processor writes location A followed by location B, any processor that sees the new value of B must also see the new value of A

TLP
Thread Level Parallelism

- Cache Coherency
  - Migration
    - Copy memory to a local cache for speed of access
    - Coherency ensures the new data is correct
  - Replication
    - Each processor can have its own copy of data
    - Reduces memory access requests

TLP
Thread Level Parallelism

- Cache Coherency Protocols
  - Directory Based
    - SMP – Central location (converged memory) used to store all sharing info
    - DSM – distributed directories
  - Snooping
    - Each cache monitors all memory activity – looks for requests to items it has in its local memory

TLP
Thread Level Parallelism

- Snooping Coherency Protocols
  - Write Invalidate
    - Processor has exclusive access to a memory value before it overwrites it
    - Invalidates all other copied prior to write
    - Use the bus structure to maintain serialization ????

Processor activity	Bus activity	Contents of processor A's cache	Contents of processor B's cache	Contents of memory location X
				0
Processor A reads X	Cache miss for X	0		0
Processor B reads X	Cache miss for X	0	0	0
Processor A writes a 1 to X	Invalidation for X	1		0
Processor B reads X	Cache miss for X	1	1	1

TLP
Thread Level Parallelism

- Snooping Coherency Protocols
  - Write Update
    - Update all copies of the memory on a write
    - Very bandwidth intensive

TLP
Thread Level Parallelism

- Snooping Coherency Protocols
  - Invalidate
    - Writing Processor
      - Broadcast an invalidate on the bus
    - All processors “snoop” on the bus
      - Watching all addresses for writes
      - Checking to see if they have a copy
      - If yes, invalidate their copy
    - In a bus system
      - Bus access limitations ensure serialization

TLP
Thread Level Parallelism

- Snooping Coherency Protocols
  - Finding data
    - Write through Caches
      - Data is always available at the common memory level
      - What about write buffers – need to treat them as still being in the cache
    - Write back Caches
      - Requested data may be in someone else's local cache
      - All processors snoop on the bus
        - When the processor with a dirty copy of the block sees the request for the block on the bus it:
          aborts the request to the higher level of memory
          provides the data to the bus (and ultimately the requester)
      - Complication
        - may take longer than just getting the block from the higher level cache
        - Multi-processor system use writeback because the improvement in bus utilization associated with WB exceeds this penalty

TLP
Thread Level Parallelism

- Snooping Coherency Protocols
  - Marking data
    - Valid and Dirty act as before
    - Exclusive
      - Indicates that only one copy exists
    - When a processor sees another copy requested it updates it's block state to shared
    - Exclusive writes do not generate invalidates
      - But they do update their own state to exclusive

TLP

Thread Level Parallelism

- Snooping Controller
  - Simple 3 state controller
    - Invalid, shared, modified

Request	Source	State of addressed cache block	Type of cache action	Function and explanation
Read hit	Processor	Shared or modified	Normal hit	Read data in local cache.
Read miss	Processor	Invalid	Normal miss	Place read miss on bus.
Read miss	Processor	Shared	Replacement	Address conflict miss: place read miss on bus.
Read miss	Processor	Modified	Replacement	Address conflict miss: write-back block, then place read miss on bus.
Write hit	Processor	Modified	Normal hit	Write data in local cache.
Write hit	Processor	Shared	Coherence	Place invalidate on bus. These operations are often called upgrade or *ownership* misses, since they do not fetch the data but only change the state.
Write miss	Processor	Invalid	Normal miss	Place write miss on bus.
Write miss	Processor	Shared	Replacement	Address conflict miss: place write miss on bus.
Write miss	Processor	Modified	Replacement	Address conflict miss: write-back block, then place write miss on bus.
Read miss	Bus	Shared	No action	Allow shared cache or memory to service read miss.
Read miss	Bus	Modified	Coherence	Attempt to share data: place cache block on bus and change state to shared.
Invalidate	Bus	Shared	Coherence	Attempt to write shared block; invalidate the block.
Write miss	Bus	Shared	Coherence	Attempt to write shared block; invalidate the cache block.
Write miss	Bus	Modified	Coherence	Attempt to write block that is exclusive elsewhere; write-back the cache block and make its state invalid in the local cache.

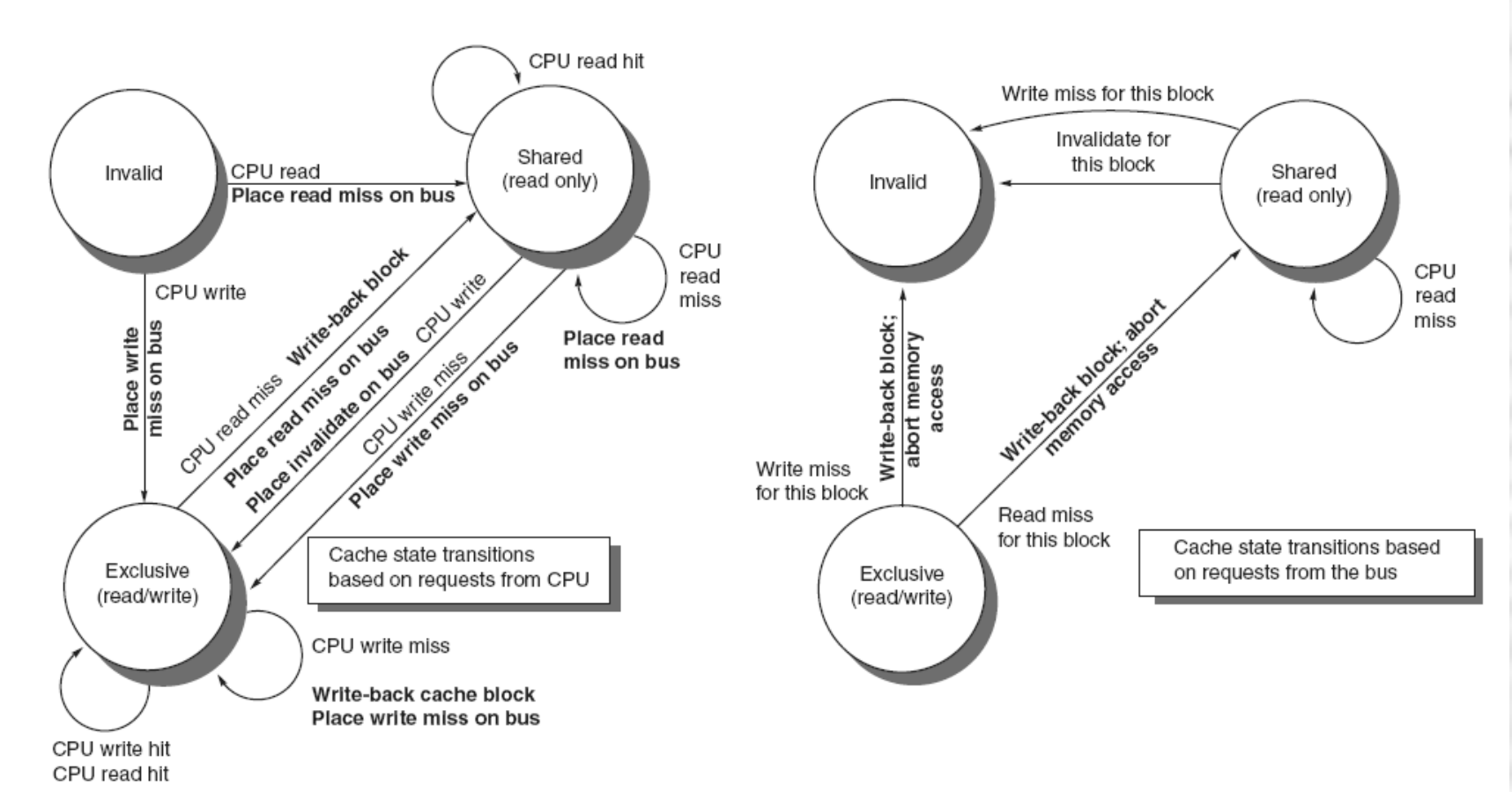
TLP
Thread Level Parallelism

- Snooping Controller
  - Simple 3 state controller
    - Invalid, shared, modified

Request	Source	State of addressed cache block	Type of cache action	Function and explanation
Read hit	Processor	Shared or modified	Normal hit	Read data in local cache.
Read miss	Processor	Invalid	Normal miss	Place read miss on bus.
Read miss	Processor	Shared	Replacement	Address conflict miss: place read miss on bus.
Read miss	Processor	Modified	Replacement	Address conflict miss: write-back block, then place read miss on bus.
Write hit	Processor	Modified	Normal hit	Write data in local cache.
Write hit	Processor	Shared	Coherence	Place invalidate on bus. These operations are often called upgrade or *ownership* misses, since they do not fetch the data but only change the state.
Write miss	Processor	Invalid	Normal miss	Place write miss on bus.
Write miss	Processor	Shared	Replacement	Address conflict miss: place write miss on bus.
Write miss	Processor	Modified	Replacement	Address conflict miss: write-back block, then place write miss on bus.
Read miss	Bus	Shared	No action	Allow shared cache or memory to service read miss.
Read miss	Bus	Modified	Coherence	Attempt to share data: place cache block on bus and change state to shared.
Invalidate	Bus	Shared	Coherence	Attempt to write shared block; invalidate the block.
Write miss	Bus	Shared	Coherence	Attempt to write shared block; invalidate the cache block.
Write miss	Bus	Modified	Coherence	Attempt to write block that is exclusive elsewhere; write-back the cache block and make its state invalid in the local cache.

TLP
Thread Level Parallelism

- Snooping Controller



TLP
Thread Level Parallelism

- Snooping Controller – issues
  - Operations are not atomic
    - “things” can happen between bus transactions
    - Create atomic bus accesses

TLP
Thread Level Parallelism

- Snooping Controller – extensions
  - Add an exclusive-clean state
    - MESI (Modified, Exclusive, Shared, Invalid)
    - On write – no invalidate generated
  - Add an owned state
    - MOESI
    - Indicates the block is in the cache and dirty
    - Used when the cache is providing updates instead of main memory