

Portland State University

ECE 588/688

Shared Memory Multiprocessors

What is a Shared Memory Architecture?

- All processors can access all memory
- Processors share memory resources, but can operate independently
- One processor's memory changes is seen by all other processors
- Easier to program
 - ◆ Communication through shared memory
 - ◆ Synchronization through locks stored in shared memory
- Need cache coherence in hardware – why?
- Need interconnection network between all processors and all memory

Shared Memory Architectures

■ **Uniform Memory Access (UMA) Architecture**

- ◆ Example: Symmetric Multiprocessor (SMP) machines
- ◆ Identical processors with equal access and equal access time to memory
- ◆ Also called CC-UMA - Cache Coherent UMA. Cache coherence means if one processor updates a location in shared memory, all the other processors know about the update

■ **Non-Uniform Memory Access (NUMA) Architecture**

- ◆ Often made by physically linking two or more SMPs
- ◆ One processor can directly access memory of another processor
- ◆ Not all processors have equal access time to all memories
- ◆ Memory access across link is slower
- ◆ Called CC-NUMA if Cache Coherence is maintained

Shared Bus Architectures

- Contention for bus and memory may degrade performance
- Need arbitration for the bus (whenever more than one bus master exists)
- Some (old) examples for shared bus architectures:
 - ◆ Encore's Multimax: Paper figure 1
 - ◆ Sequent Balance: Paper figure 2
 - ◆ Alliant FX/80: Paper figure 3
 - ◆ ELXSI System 6400: Paper figure 4

Network Multiprocessors

- More scalable than shared bus architectures
 - ◆ Less contention for shared interconnection resources
- Usually higher latency to communicate
- May need arbitration to access shared memory (if more than one processor requests access to same bank)
- Some (old) examples:
 - ◆ BBN Butterfly: Paper figure 5, 6
 - ◆ Intel iPSC/2
 - ◆ NCUBE/n
 - ◆ FPS T Series

Interconnection Networks

- In a shared memory MP, we need to connect different processors and memory modules
- Types of interconnect:
 - ◆ Shared bus
 - ◆ Crossbar: Fully connected
 - ◆ Ring
 - ◆ Mesh
 - ◆ 2-D Torus
 - ◆ Hypercube
- Number of hops vs. number of links: Compare N processors and M memory modules
- More details later in the course

Memory Hierarchy

- Problem: sharing memory means more than one processor can send requests to memory
 - ◆ High memory bandwidth requirements
- To avoid sending lots of memory requests, processors use caches to:
 - ◆ Filter out many memory requests
 - ◆ Reduce average memory latency
 - ◆ Reduce memory bandwidth requirements
- Typically more than one level of caches is used
 - ◆ L1 caches: Usually Split I & D caches, small and fast
 - ◆ L2 caches: Usually on die, composed of SRAM cells

Cache Coherence

- Problem: Using caches means multiple copies of the same memory location may exist
 - ◆ Updates to the same location may lead to bugs

- Example:

Processor 1 reads A

Processor 2 reads A

Processor 1 writes to A

Now, processor 2's cache contains stale data

- Cache coherence need to be implemented in hardware using a cache coherence protocol

Conditions for Cache Coherence

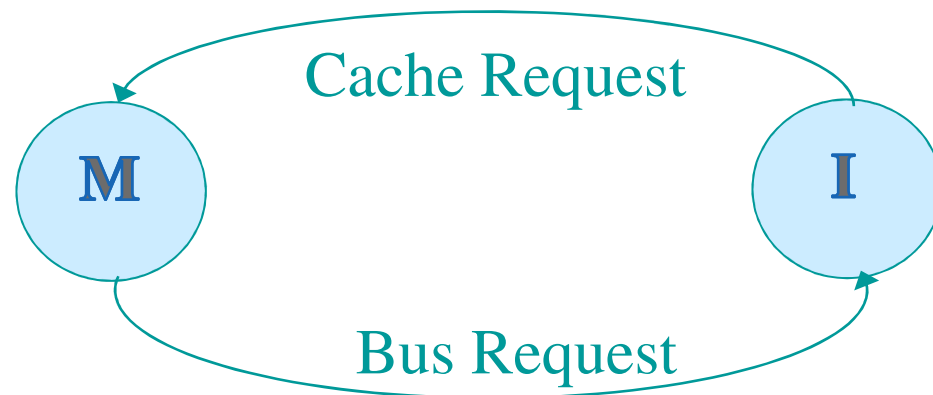
- **Program Order.** A read by processor P to location A that follows a write by P to A, with no writes to A by another processor in between, should always return the value of A written by P
- **Coherent View of Memory.** A read by processor P1 to location A that follows a write by another processor P2 to location A should return the written value by P2 if:
 - ◆ The read and write are sufficiently separated in time
 - ◆ No other writes to A by another processor occur between the read and the write
- **Write Serialization.** Writes to the same location are serialized: Two writes to the same location by any two processors are seen in the same order by all processors

Cache Coherence (Cont.)

- Cache coherence defines behavior of reads and writes to the same memory location
- Memory consistency models define the behavior of reads and writes with respect to accesses to other memory locations (More details later in the course)
- Two main types of cache coherence protocols:
 - ◆ Snooping
 - Caches keep track of the sharing status of all blocks
 - No centralized state is kept
 - Cache controllers snoop shared interconnect to know when a requested block exists in the cache
 - ◆ Directory
 - Sharing status of any block in memory is kept in one location

Very Simple Coherence Protocol

- MI protocol
 - ◆ Two states: M (Modified) and I (Invalid)
 - ◆ Only one cache contains a copy of a certain memory location
 - ◆ When another cache requests a block, the cache currently containing the block invalidates it
 - ◆ Protocol limits sharing and degrades performance



- Optimization: MSI protocol allows read sharing

Reading Assignment

- Per Stenstrom, "A Survey of Cache Coherence Schemes for Multiprocessors," IEEE Computer, 1990 (Review)
- Project proposals due Friday
 - ◆ Discuss project information