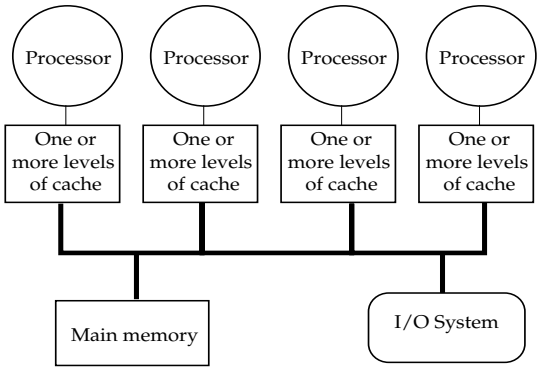
Small-Scale MIMD Designs

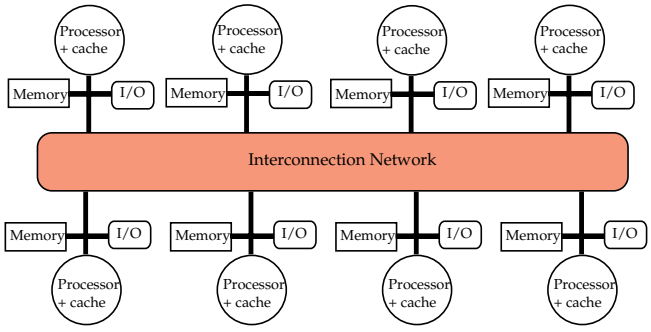
- **Memory: centralized with uniform access time and bus interconnect**
- **Examples: SPARCCenter, Challenge, SystemPro**



MPs.intro 1

Large-Scale MIMD Designs

- **Memory: distributed with nonuniform access time and scalable interconnect (distributed memory)**
- **Examples: T3D, Exemplar, Paragon, CM-5**



MPs.intro 2

Communication Models

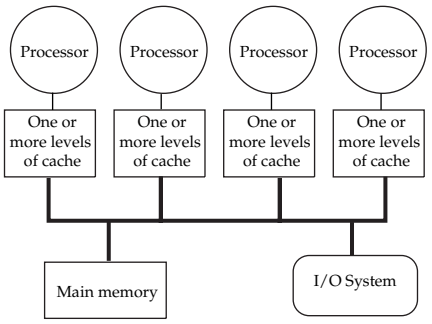
- **Shared Memory**
  - **processors communicate with shared address space**
  - **easy on small-scale machines**
  - **Advantages:**
    - » **model of choice for uniprocessors, small-scale MPs**
    - » **ease of programming**
    - » **lower latency**
    - » **easier to use hardware controlled caching**
- **Message passing**
  - **processors have private memories communicate with messages**
  - **Advantages:**
    - » **less hardware, easier to design**
    - » **focuses attention on costly operations**
- **Can support either model on either HW base**

Important Communication Properties

- **Bandwidth**
  - **need high bandwidth in communication**
  - **cannot scale, but stay close**
  - **make limits in network, memory, and processor**
  - **overhead to communicate is a problem in many machines**
- **Latency**
  - **affects performance, since processor may have to wait**
  - **affects ease of programming, since requires more thought to overlap communication and computation**
- **Latency hiding**
  - **how can a mechanism help hide latency**
  - **Examples: overlap message send with computation, prefetch**

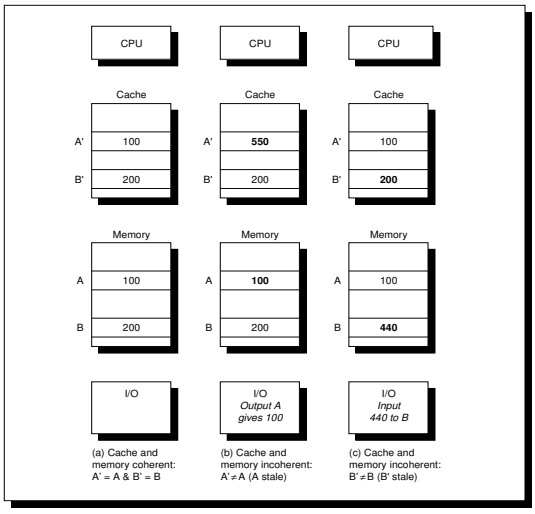
Small-Scale, Shared Memory

- **Caches serve both to**
  - **increase bandwidth versus bus/memory**
  - **reduce latency of access**
  - **valuable for both private data and shared data**



MPs.intro 5

The Problem of Cache Coherency



MPs.intro 6

What does coherency mean?

- **Informally:**
  - **any read must return the most recent write (too strict)**
- **Better:**
  - **any write must eventually be seen by a read**
  - **all writes are seen in order**
- **Two rules to ensure this:**
  - **If P writes x and P1 reads it, P's write will be seen if the read and write are sufficiently far apart.**
  - **Writes to a single location are serialized: seen in one order**
    - » **latest write will be seen**
    - » **could see writes in illogical order (could see older value after a newer value)**

MPs.intro 7

Potential Solutions

- **Snooping solution(snoopy bus):**
  - **Send all requests for data to all processors.**
  - **Processors snoop to see if they have a copy and respond accordingly**
  - **requires broadcast, since caching information is at processors**
  - **works well with bus**
- **Directory based schemes**
  - **keep track of what is being shared in one place**
  - **distributed memory => distributed directory**
  - **send point-to-point requests to processors**

MPs.intro 8

Basic Snoopy Protocols

- **Write Invalidate protocol:**
  - **Write to shared data: an invalidate is sent to all caches which snoop and *invalidate* any copies**
  - **Read miss:**
    - » **write-through: memory is always up-to-date**
    - » **write-back: snoop in caches to find most recent copy**
- **Write Broadcast Protocol:**
  - **Write to shared data: broadcast on bus, processors snoop, and *update* copies**
  - **Read miss: memory is always up-to-date**
- **Write serialization: bus serializes requests**

Basic Snoopy Protocols

- **Write invalidate versus broadcast:**
  - **invalidate requires one transaction per write-run**
  - **invalidate uses spatial locality: one transaction per block**
  - **broadcast has lower latency between write and read**
  - **BW vs. latency tradeoff**

Name	Protocol Type	Memory-write policy	Machines using
Write Once	Write invalidate	Write back after first write	First snoopy protocol.
Synapse N+1	Write invalidate	Write back	1st cache-coherent MPs
Berkeley	Write invalidate	Write back	Berkeley SPUR .
Illinois	Write invalidate	Write back	SGI Power and Challenge
"Firefly"	Write broadcast	Write back private, Write through shared	SPARCCenter 2000

An Example Snoopy Protocol

- **Invalidation protocol, write-back cache**
- **Each block of memory is in one state:**
  - **clean in all caches and up-to-date in memory**
  - **dirty in exactly one cache**
  - **now in any caches**
- **Each cache block is in one state:**
  - **Invalid: block contains no data**
  - **Shared: block can be read**
  - **Exclusive: cache has only copy, its writeable, and dirty**
- **Read misses: cause all caches to snoop**
- **Writes to clean line are treated as misses**