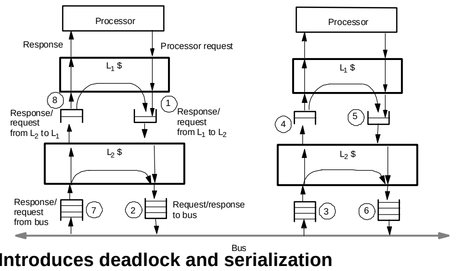
Case Studies

CS 258, Spring 99
David E. Culler
Computer Science Division
U.C. Berkeley

Multi-Level Caches with ST Bus

Key new problem: many cycles to propagate through hierarchy
- Must let others propagate too for bandwidth, so queues between levels



- **Introduces deadlock and serialization problems**

Deadlock Considerations

- **Fetch deadlock:**
  - Must buffer incoming requests/responses while request outstanding
  - One outstanding request per processor => need space to hold *p* requests plus one reply (latter is essential)
  - If smaller (or if multiple o/s requests), may need to NACK
  - Then need priority mechanism in bus arbiter to ensure progress
- **Buffer deadlock:**
  - L1 to L2 queue filled with read requests, waiting for response from L2
  - L2 to L1 queue filled with bus requests waiting for response from L1
  - Latter condition only when cache closer than lowest level is write back
  - Could provide enough buffering, or general solutions discussed later
- **If # o/s bus transactions smaller than total o/s cache misses, response from cache must get bus before new requests from it allowed**
  - Queues may need to support bypassing

Sequential Consistency

- **Separation of commitment from completion even greater now**
  - More performance-critical that commitment replace completion
- **Fortunately techniques for single-level cache and ST bus extend**
  - Just use them at each level
  - i.e. either don't allow certain reorderings of transactions at any level
  - Or don't let outgoing operation proceed past level before incoming invalidations/updates at that level are applied

Multiple Outstanding Processor Requests

- **So far assumed only one: not true of modern processors**
- **Danger: operations from same processor can complete out of order**
  - e.g. write buffer: until serialized by bus, should not be visible to others
  - Uniprocessors use write buffer to insert multiple writes in succession
    - » multiprocessors usually can't do this while ensuring consistent serialization
    - » exception: writes are to same block, and no intervening ops in program order
- **Key question: who should wait to issue next op till previous completes**
  - Key to high performance: processor needn't do it (so can overlap)
  - Queues/buffers/controllers can ensure writes not visible to external world and reads don't complete (even if back) until allowed (more later)
- **Other requirement: caches must be lockup free to be effective**

Case Studies of Bus-based Machines

- **SGI Challenge, with Powerpath bus**
- **SUN Enterprise, with Gigaplane bus**
  - Take very different positions on the design issues discussed above
- **Overview**
- **For each system:**
  - Bus design
  - Processor and Memory System
  - Input/Output system
  - Microbenchmark memory access results
- **Application performance and scaling (SGI Challenge)**

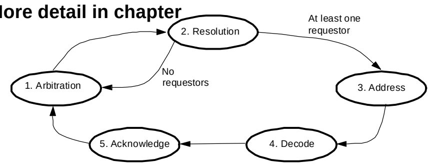
NOW Handout Page 1

Bus Design Issues

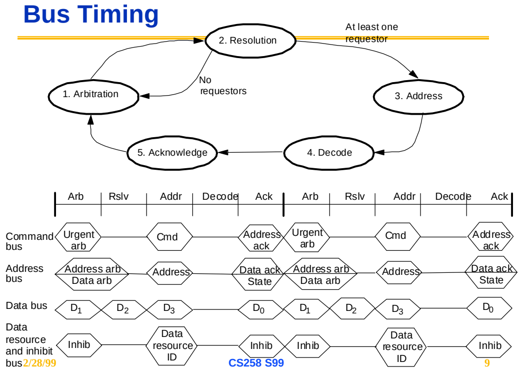
- Multiplexed versus non-multiplexed (separate addr and data lines)
- Wide versus narrow data busses
- Bus clock rate
  - Affected by signaling technology, length, number of slots...
- Split transaction versus atomic
- Flow control strategy

SGI Powerpath-2 Bus

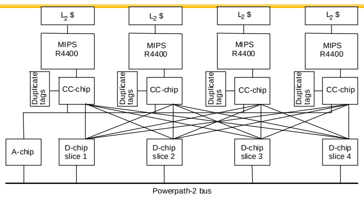
- Non-multiplexed, 256-data/40-address, 47.6 MHz, 8 o/s requests
- Wide => more interface chips so higher latency, but more bw at slower clock
- Large block size also calls for wider bus
- Uses Illinois MESI protocol (cache-to-cache sharing)
- More detail in chapter



Bus Timing



Processor and Memory Systems

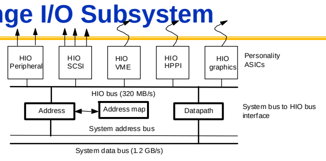


- 4 MIPS R4400 processors per board share A and D chips
- A chip has address bus interface, request table, control logic
- CC chip per processor has duplicate set of tags
- Processor requests go from CC chip to A chip to

Memory Access Latency

- 250ns access time from address on bus to data on bus
- But overall latency seen by processor is 1000ns!
  - 300 ns for request to get from processor to bus
    - » down through cache hierarchy, CC chip and A chip
  - 400ns later, data gets to D chips
    - » 3 bus cycles to address phase of request transaction, 12 to access main memory, 5 to deliver data across bus to D chips
  - 300ns more for data to get to processor chip
    - » up through D chips, CC chip, and 64-bit wide interface to processor chip, load data into primary cache, restart pipeline

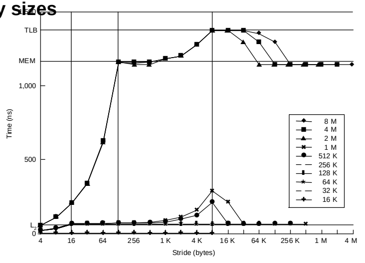
Challenge I/O Subsystem



- Multiple I/O cards on system bus, each has 320MB/s HIO bus
  - Personality ASICs connect these to devices (standard and graphics)
- Proprietary HIO bus
  - 64-bit multiplexed address/data, same clock as system bus
  - Split read transactions, up to 4 per device
  - Pipelined, but centralized arbitration, with several transaction lengths
  - Address translation via mapping RAM in system bus interface
- Why the decouplings? (Why not connect directly to system bus?)

Challenge Memory System Performance

- Read microbenchmark with various strides and array sizes

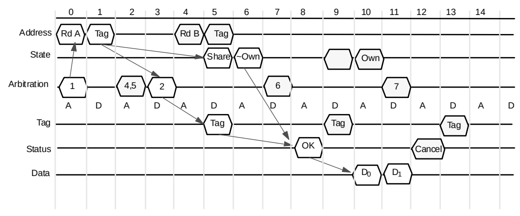


Ping-pong flag-spinning microbenchmark: round-trip time 6.2 µs.

Sun Gigaplane Bus

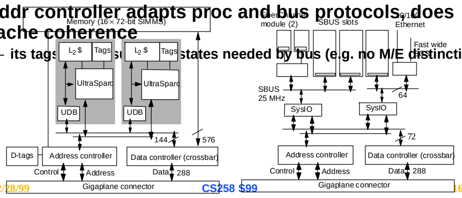
- Non-multiplexed, split-transaction, 256-data/41-address, 83.5 MHz
  - Plus 32 ECC lines, 7 tag, 18 arbitration, etc. Total 388.
- Cards plug in on both sides: 8 per side
- 112 outstanding transactions, up to 7 from each board
  - Designed for multiple outstanding transactions per processor
- Emphasis on reducing latency, unlike Challenge
  - Speculative arbitration if address bus not scheduled from prev. cycle
  - Else regular 1-cycle arbitration, and 7-bit tag assigned in next cycle
- Snoop result associated with request phase (5 cycles later)
- Main memory can stake claim to data bus 3 cycles into this, and start memory access speculatively
  - Two cycles later, asserts tag bus to inform others of coming transfer
- MOESI protocol (owned state for cache-to-cache

Gigaplane Bus Timing



Enterprise Processor and Memory System

- 2 procs per board, external L2 caches, 2 mem banks with x-bar
- Data lines buffered through UDB to drive internal 1.3 GB/s UPA bus
- Wide path to memory so full 64-byte line in 1 mem cycle (2 bus cyc)
- Addr controller adapts proc and bus protocols, does cache coherence
  - its tags [illegible] states needed by bus (e.g. no M/E distinction)

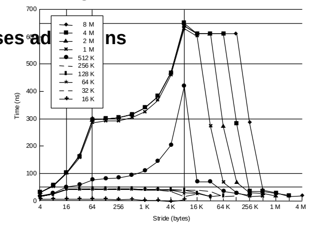


Enterprise I/O System

- I/O board has same bus interface ASICs as processor boards
- But internal bus half as wide, and no memory path
- Only cache block sized transactions, like processing boards
  - Uniformity simplifies design
  - ASICs implement single-block cache, follows coherence protocol
- Two independent 64-bit, 25 MHz Sbuses
  - One for two dedicated FiberChannel modules connected to disk
  - One for Ethernet and fast wide SCSI
  - Can also support three SBUS interface cards for arbitrary peripherals
- Performance and cost of I/O scale with no. of I/O

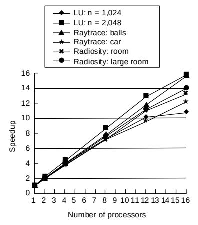
Memory Access Latency

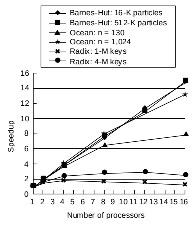
- 300ns read miss latency
- 11 cycle min bus protocol at 83.5 Mhz is 130ns of this time
- Rest is path through caches and the DRAM access
- TLB misses ad[illegible]ns



Ping-pong microbenchmark is 1.7 µs round-trip (5 mem accesses)

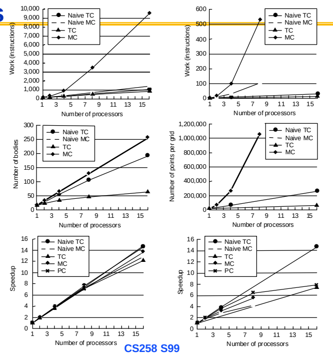
Application Speedups (Challenge)





- **Problem in Ocean with small problem: communication and barrier cost**
- **Problem in Radix: contention on bus due to very high traffic**
  - **also leads to high imbalances and barrier wait time**

Application Scaling under Other Models



NOW Handout Page 4