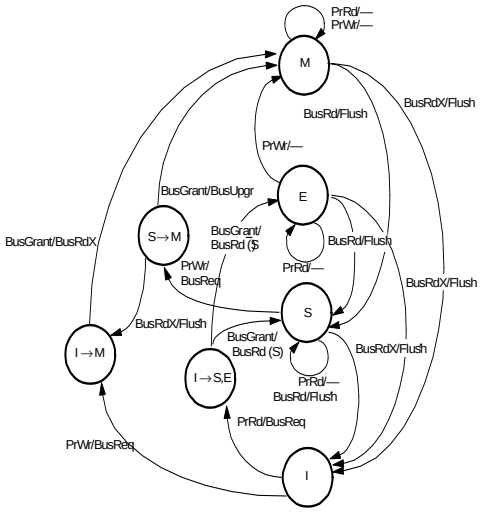
18-742

Lecture 10

Symmetric Multiprocessors II

Spring 2005
Prof. Babak Falsafi
http://www.ece.cmu.edu/~ece742



Slides developed in part by Profs. Adve, Falsafi, Hill, Lebeck, Reinhardt, Smith, and Singh of University of Illinois, Carnegie Mellon University, University of Wisconsin, Duke University, University of Michigan, and Princeton University.

Announcements

Homework 5 is posted

Due next Wednesday

Talk at Intel this Friday

Can Parallel Computing Finally Impact Mainstream Computing?

Uzi Vishkin, University of Maryland

(10:30am @ Intel Research Pittsburgh)

Readings

Chapter 6 of the book

Reader 4

- **S. L. Scott, *Synchronization and Communication in the T3E Multiprocessor*, ISCA 1996.**
- **BlueGene/L Team, *An Overview of the BlueGene/L SuperComputer*, SC 2002: 1-22.**

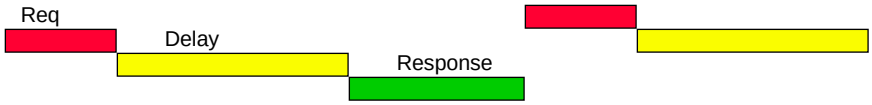
Shared Caches

- **Share low level caches among multiple processors**
  - **Sharing L1 adds to latency, *unless* multithreaded processor**
- **Advantages**
  - **Eliminates need for coherence protocol at shared level**
  - **Reduces latency within sharing group**
  - **Processors essentially prefetch for each other**
  - **Can exploit working set sharing**
  - **Increases utilization of cache hardware**
- **Disadvantages**
  - **Higher bandwidth requirements**
  - **Increased hit latency**
  - **May be more complex design**
  - **Lower effective capacity if working sets don't overlap**
- **Bottom Line**
  - **Packaging has a lot to do with it**
  - **As levels of integrations increase, there will be more sharing**

Split-transaction (Pipelined) Bus

- **Supports multiple simultaneous transactions (many designs)**

Atomic Transaction Bus



Split-transcation Bus

Potential Problems

- **Two transactions to same block (conflicting)**
  - **Mid-transaction snoop hits**
- **Buffer requests and responses**
  - **Need flow control to prevent deadlock**
- **Ordering of Snoop responses**
  - **when does snoop response appear wrt data response**

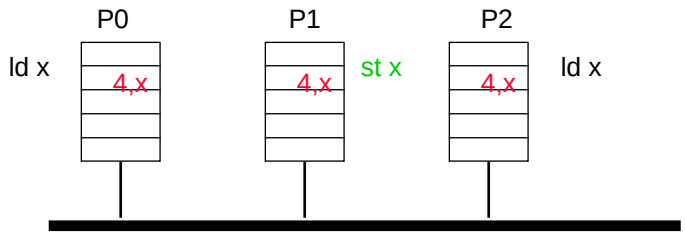
One Solution

- NACK for flow control
- Out-of-order responses
  - snoop results presented with data response
- Disallow conflicting transactions

A Split-transaction Bus Design

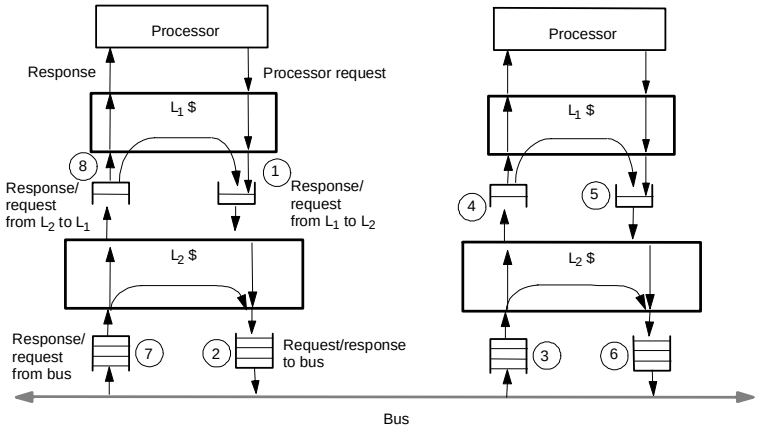
- 4 Buses + Flow Control and Snoop Results
  - Command (type of xaction)
  - Address
  - Tag (unique identifier for response)
  - Data (doesn't require address)
- Form of transactions
  - BusRD, BusRDX (request + response)
  - Writeback (request + data)
  - Upgrade (request only)
- Per Processor Request Table Tracks All Transactions

A Simple Example



P2 Can snoop data from first ld
P1 Must hold st operation until entry is clear

Multi-Level Caches with Split Bus

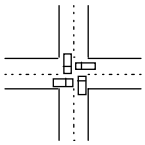


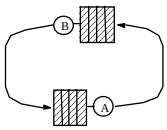
Multi-level Caches with Split-Transaction Bus

- General structure uses queues between
  - Bus and L2 cache
  - L2 cache and L1 cache
- Deadlock!
- Classify all transactions
  - Request, only generates responses
  - Response, doesn't generate any other transactions
- Requestor guarantees space for all responses
- Use Separate Request and Response queues

More on Correctness

- Partial correctness (never wrong): Maintain coherence and consistency
- Full correctness (always right): Prevent:
- Deadlock:
  - all system activity ceases
  - Cycle of resource dependences





- Livelock:
  - no processor makes forward progress
  - constant on-going transactions at hardware level
  - e.g. simultaneous writes in invalidation-based protocol
- Starvation:
  - some processors make no forward progress
  - e.g. interleaved memory system with NACK on bank busy

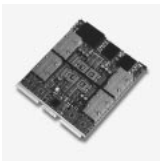
Deadlock, Livelock, Starvation

- **Request-reply protocols can lead to *deadlock***
  - **When issuing requests, must service incoming transactions**
  - **e.g. cache awaiting bus grant must snoop & flush blocks**
  - **else may not respond to request that will release bus: deadlock**
- **Livelock:**
  - **window of vulnerability problem [Kubi et al., MIT]**
  - **Handling invalidations between obtaining ownership & write**
  - **Solution: don't let exclusive ownership be stolen before write**
- **Starvation:**
  - **solve by using fair arbitration on bus and FIFO buffers**

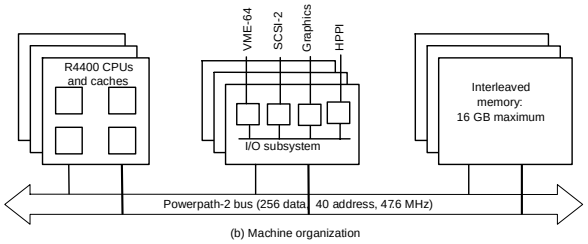
Deadlock Avoidance

- **Responses are never delayed by requests waiting for a response**
- **Responses are guaranteed to be sunk**
- **Requests will eventually be serviced since the number of responses is bounded by outstanding requests**
- **Must classify transactions according to deadlock and coherence semantics**
  - **e.g., ordering of BusRD response (Bdata) and BInval**
  - **Treat both Bdata and Binval as requests (go in same queue)**

SGI Challenge Overview



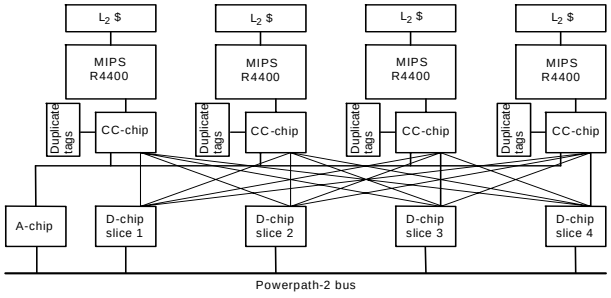
(a) A four-processor board



(b) Machine organization

- **36 MIPS R4400 (peak 2.7 GFLOPS, 4 per board) or 18 MIPS R8000 (peak 5.4 GFLOPS, 2 per board)**
- **8-way interleaved memory (up to 16 GB)**
- **1.2 GB/s Powerpath-2 bus @ 47.6 MHz, 16 slots, 329 signals**
- **128 Bytes lines (1 + 4 cycles)**
- **Split-transaction with up to 8 outstanding reads**
  - **all transactions take five cycles**
- **Miss latency nearly 1 us (mostly on CPU board, not bus…)**

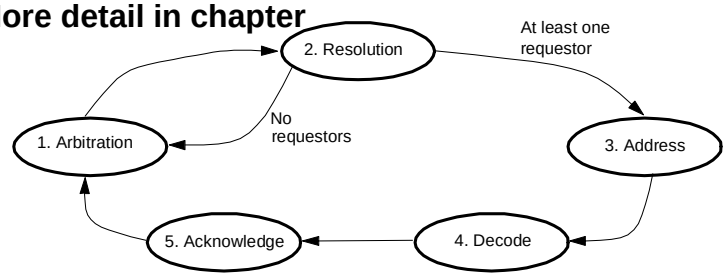
Processor and Memory Systems



- **4 MIPS R4400 processors per board share A / 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 bus**
- **4 bit-sliced D chips interface CC chip to bus**

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

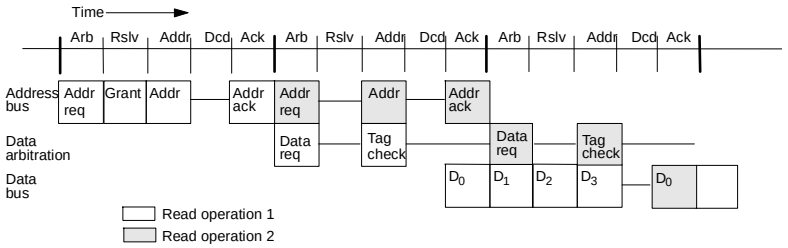


Bus Design and Req-Resp Matching

- Essentially two separate buses, arbitrated independently
  - "Request" bus for command and address
  - "Response" bus for data
- Out-of-order responses imply need for matching req-response
  - Request gets 3-bit tag when wins arbitration (8 outstanding max)
  - Response includes data as well as corresponding request tag
  - Tags allow response to not use address bus, leaving it free
- Separate bus lines for arbitration, and for snoop results

Bus Design (continued)

- **Each of request and response phase is 5 bus cycles**
  - **Response: 4 cycles for data (128 bytes, 256-bit bus), 1 turnaround**
  - **Request phase: arbitration, resolution, address, decode, ack**
  - **Request-response transaction takes 3 or more of these**



Cache tags looked up in decode; extend ack cycle if not possible

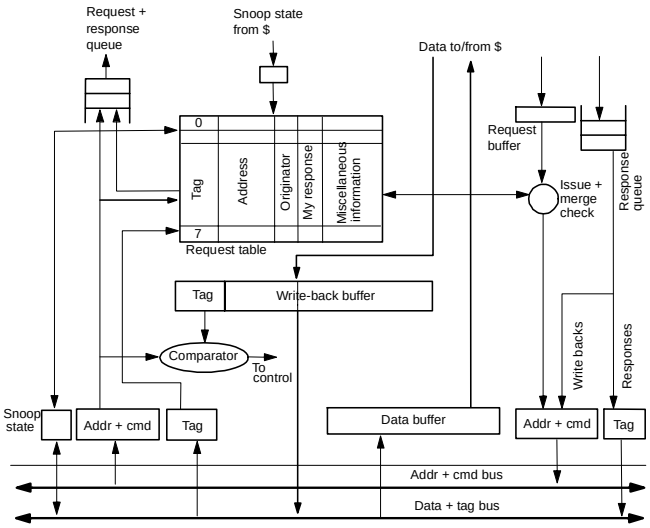
- Determine who will respond, if any
- Actual response comes later, with re-arbitration

Write-backs have request phase only: arbitrate both data+addr buses

Bus Design (continued)

- **Flow-control through *negative acknowledgement (NACK)***
- **No conflicting requests for same block allowed on bus**
  - **8 outstanding requests total, makes conflict detection tractable**
  - **Eight-entry "request table" in each cache controller**
  - **New request on bus added to all at same index, determined by tag**
  - **Entry holds address, request type, state in that cache (if determined already), ...**
  - **All entries checked on bus or processor accesses for match, so fully associative**
  - **Entry freed when response appears, so tag can be reassigned by bus**

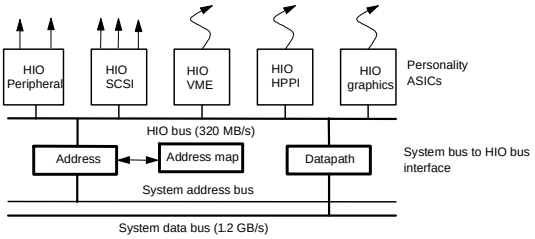
Bus Interface with Request Table



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**

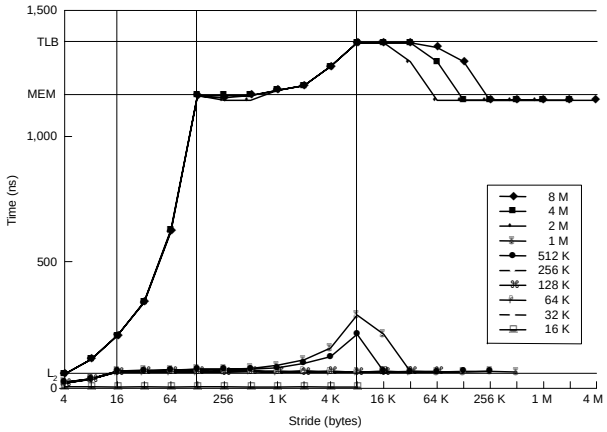
Challenge I/O Subsystem



- **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, split read trans., up to 4 per device**
  - **Pipelined, but centralized arbitration, with several transaction lengths**
  - **Address translation via mapping RAM in system bus interface**
- **I/O board acts like a processor to memory system**

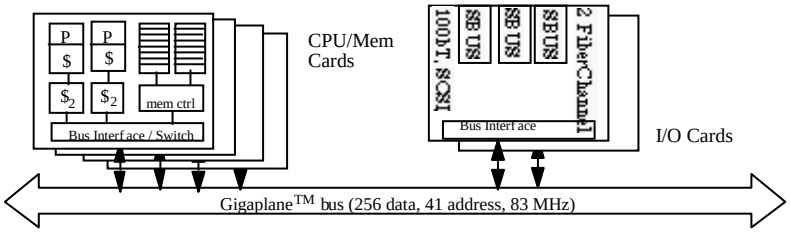
Challenge Memory System Performance

- **Read microbenchmark w/ various strides / array sizes**



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

SUN Enterprise 6000 Overview

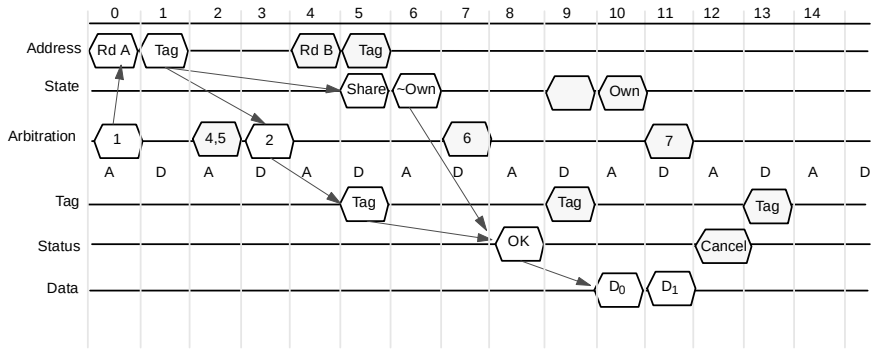


- **Up to 30 UltraSPARC processors, MOESI protocol**
- **Gigaplane™ bus has peak bw 2.67 GB/s, 300 ns latency**
- **Up to 112 outstanding transactions (max 7 per board)**
- **16 bus slots, for processing or I/O boards**
  - **2 CPUs and 1GB memory per board**
    - » **memory distributed, but protocol treats as centralized (UMA)**

Sun Gigaplane Bus

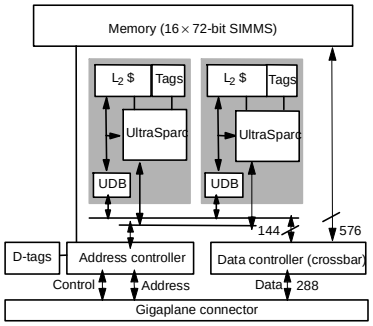
- **Non-multiplexed, split-transaction, 256-data/41-address, 83.5 MHz (Plus 32 ECC lines, 7 tag, 18 arbitration, etc. Total 388)**
- **Cards plug in on both sides: 8 per side**
- **112 outstanding transactions, up to 7 from each board**
  - **Designed for multiple outstanding transactions per processor**
- **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 (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**

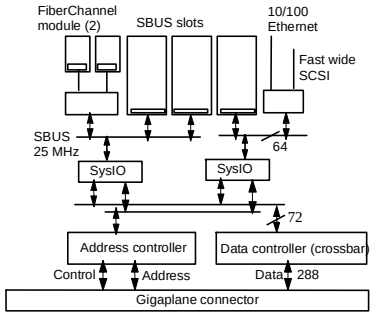
Gigaplane Bus Timing



Enterprise Processor and Memory System

- **2 procs / board, ext. L2 caches, 2 mem banks w/ 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 2 bus cycles**



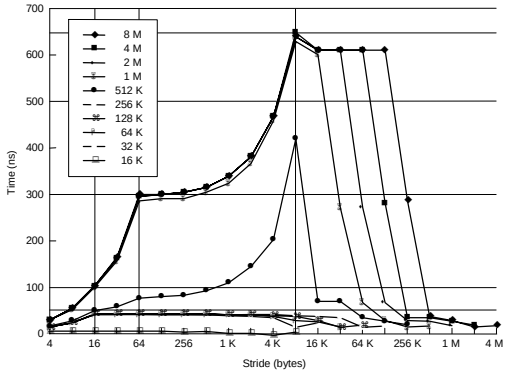


Enterprise I/O System

- 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 boards

Memory Access Latency

- 300ns read miss latency (130 ns on bus)
- Rest is path through caches & the DRAM access
- TLB misses add 340 ns



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

Sun Enterprise 10000

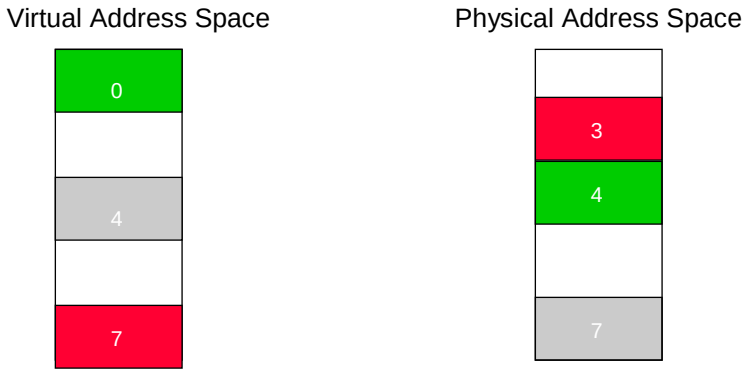
- How far can you go with snooping coherence?
- Quadruple request/snoop bandwidth using four address busses
  - each handles 1/4 of physical address space
  - impose *logical* ordering for consistency: for writes on same cycle, those on bus 0 occur "before" bus 1, etc.
- Get rid of data bandwidth problem: use a network
  - E10000 uses 16x16 crossbar betw. CPU boards & memory boards
  - Each CPU board has up to 4 CPUs: max 64 CPUs total
- 10.7 GB/s max BW, 468 ns unloaded miss latency
- See "Starfire: Extending the SMP Envelope", IEEE Micro, Jan/Feb 1998

Outline

- Coherence Control Implementation
- Writebacks, Non-Atomicity, & Serialization/Order
- Hierarchical Cache
- Split Buses
- Deadlock, Livelock, & Starvation
- Three Case Studies
- TLB Coherence
- Virtual Cache Issues

Translation Lookaside Buffer

- **Cache of Page Table Entries**
- **Page Table Maps Virtual Page to Physical Frame**



The TLB Coherence Problem

- **Since TLB is a cache, must be kept coherent**
- **Change of PTE on one processor must be seen by all processors**
- **Process migration**
- **Changes are infrequent**
  - **get OS to do it**
  - **Always flush TLB is often adequate**

TLB Shootdown

- To modify TLB entry, modifying processor must
  - LOCK page table,
  - flush TLB entries,
  - queue TLB operations,
  - send interprocessor interrupt,
  - spin until other processors are done
  - UNLOCK page table
- SLOW...
  - But most common solution today
- Some ISAs have “flush TLB entry” instructions

Virtual Caches & Synonyms

- Problem
  - Synonyms: V0 & V1 map to P1
  - When doing coherence on block in P1 how do you find V0 & V1?
- Don’t do virtual caches (most common today)
- Don’t allow synonyms
  - Constrains software (and OS assumptions)
- Allow virtual cache & synonyms
  - How implement reverse address translation?
  - See Wang et al. next

Wang et al. [ISCA89]

- **Basic Idea**
  - Virtual L1 and physical L2
  - Do coherence on physical addresses
  - Each L2 block maintains backpointer to corresponding L1 block (if any)
    (requires log2 #L1_blocks - log2 (page_size / block_size)
  - Never allow block to be simultaneously cached under synonyms
- **Example where V0 & V1 map to P2**
  - Initially V1 in L1 and P2 in L1 points to V1
  - Processor references V0
  - L1 miss
  - L2 detects synonym in L1
  - Change L1 tag and L2 pointer so that L1 has V0 instead of V1
  - Resume

Virtual Caches & Homonyms

- **Homonym**
  - V0 of one process maps to P2, while V0 of other process maps to P3
- **Flush cache on context switch**
  - simple but performs poorly
- **Address-space IDs (ASIDs)**
  - in architecture & part of context state

Outline

- **Coherence Control Implementation**
- **Writebacks, Non-Atomicity, & Serialization/Order**
- **Hierarchical Cache**
- **Split Buses**
- **Deadlock, Livelock, & Starvation**
- **Three Case Studies**
- **TLB Coherence**
- **Virtual Cache Issues**