CS/ECE 757: Advanced Computer Architecture II
(Parallel Computer Architecture)

Symmetric Multiprocessors Part 2 (Case Studies)

Copyright 2001 Mark D. Hill
University of Wisconsin-Madison

Slides are derived from work by
Sarita Adve (Illinois), Babak Falsafi (CMU),
Aly Lebeck (Duke), Steve Reinhardt (Michigan),
and J. P. Singh (Princeton). Thanks!

SGI Challenge Overview

- 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...)

(C) 2001 Mark D. Hill from Adve,Falsafi, Lebeck, Reinhardt, Singh CS/ECE 757
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
  large block size also calls for wider bus
- Processor requests go from CC to A chip to bus
- A chip has address bus interface, request table
- Processor has duplicate set of tags
- 4 bit-sliced D chips interface CC chip to bus

1. Arbitration
2. Resolution
3. Address
4. Decode
5. Acknowledge

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
Bus Design (continued)

- 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

![Diagram](image)

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

<table>
<thead>
<tr>
<th>Address</th>
<th>State</th>
<th>Arbitration</th>
<th>Tag</th>
<th>Status</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>R/A</td>
<td>Tag</td>
<td>1</td>
<td>A</td>
<td>D</td>
<td>A</td>
</tr>
<tr>
<td>R/B</td>
<td>Tag</td>
<td>4,5</td>
<td>D</td>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>A</td>
<td>D</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6</td>
<td>A</td>
<td>D</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7</td>
<td>A</td>
<td>D</td>
<td>A</td>
</tr>
</tbody>
</table>

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?

---

**Graph**

```
Memory bandwidth
Snooping capacity
Data-crossbar capacity with random addresses
```

Bytes per clock vs. System boards

<table>
<thead>
<tr>
<th>System boards</th>
<th>Bytes per clock</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>64</td>
</tr>
<tr>
<td>3</td>
<td>128</td>
</tr>
<tr>
<td>4</td>
<td>192</td>
</tr>
<tr>
<td>5</td>
<td>256</td>
</tr>
<tr>
<td>16</td>
<td>21,333</td>
</tr>
</tbody>
</table>

Bandwidth at 63.3 MHz clock (Mbps)

- 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 between 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
Sun Enterprise 10000: Read Miss

- Send Address and Establish Coherency

1. Processor makes a request to its port controller.
2. Port controller sends the address to a coherency interface controller, and sends the request to the local address arbiter.
3. Local address arbiter requests a global address bus cycle.
4. Global address arbiter grants an address bus cycle.
5. Coherency interface controller sends the address through the global address bus to the rest of the coherency interface controllers on the other boards.
6. All coherency interface controllers relay the address to their memory controllers and snoop the address in their duplicate tags.
7. All coherency interface controllers send their snoop results to the global address arbiter.
8. Global address arbiter broadcasts the global snoop result.
9. Memory is not aborted by its coherency interface controller because the snoop did not hit.
Sun Enterprise 10000: Read Miss

- **Read from memory**
  1. Memory controller recognizes that this address is for one of its memory banks.
  2. Memory controller orchestrates a DRAM cycle and requests a data transfer from its local data arbiter.
  3. Memory sends 72 bytes of data to the unpack unit.
  4. Unpack splits the data into four 18-byte pieces.
  5. Unpack sends data to the data buffer to be buffered for transfer.

Sun Enterprise 10000: Read Miss

- **Transfer Data**
  1. Local data arbiter requests a data transfer.
  2. Global data arbiter grants a data transfer and notifies the receiving local data arbiter that data is coming.
  3. Sending local data arbiter tells the data buffer to begin the transfer.
  4. Sending data buffer sends data to the local data router.
  5. Data moves through the local data router to the centerplane crossbar.
  6. Data moves through the centerplane crossbar to the receiving board’s local data router.
  7. Data moves through the receiving local data router to the receiver’s data buffer.
Sun Enterprise 10000: Read Miss

- **Write Data**
  1. Port controller tells the data buffer to send the data packet.
  2. Data buffer sends data to the UltraSparc data buffer on the processor module.
  3. UltraSparc data buffer sends data to the processor.

Piranha: A Scalable CMP

- **Commercial workloads**
  - Large instruction and data footprints
    - execution often dominated by memory stalls
  - Parallelism arises from independent instruction streams
  - Multiple out of order issue offer small gains for OLTP
  - No floating point or multimedia
  - Complex desktop processors may not be compatible with server applications

- **Server chips**
  - Simultaneous Multithreading (SMT)
    - complex core shared by multiple contexts
  - Chip Multiprocessors (CMP)
    - multiple simple cores + shared caches
Piranha: A Scalable CMP

- **Single Die contains:**
  - 8 simple alpha processor cores
    - single issue, in-order
    - 8 stage pipelines
  - Separate L1 data/inst caches
    - 64KB 2-way L1 caches
      - MESI protocol
  - Shared L2 cache
    - 1MB
    - 8 banks
    - 8-way per bank
      - no-inclusion
  - 8 memory controllers
    - section of memory controlled by each chip
  - 2 coherence engines
  - network router

![Block diagram of a single-chip Piranha processing node.](image)

On-chip Caches

- **Non-Inclusion**
  - Copy of L1 tags in L2 controller
  - L1 misses that also miss in L2
    - fill directly from main memory
    - no copy in L2
  - L2 filled only when L1 line is replaced
    - L2 acts as large victim cache
    - Even clean L1 replacements write back to L2
      - One L1 is owner (exclusive or last requester)
      - writeback only when owner L1 replaces data
On-chip Caches

- **Intra-Chip coherence**
  - L2 controllers contain all on-chip sharing information
  - Both L1 and L2 tags checked on L2 accesses
  - Memory request from L1 sent to L2
  - L2 then:
    - services request directly
    - forwards request to owner L1
    - forwards request to protocol engine (needs to go off chip)
    - obtain data from local section of memory

Piranha Evaluation

- **Benchmarks**
  - OLTP – like TPC-B
  - Decision Support – like TPC-D
  - Based on Oracle

- **Simulated Architectures:**
  - consider both simple and advanced technologies
    (500 MHz and 1 GHz)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Piranha (PR)</th>
<th>Next-Generation Microprocessor (O000)</th>
<th>Full-Custom Piranha (PR)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processor Speed</td>
<td>500 MHz</td>
<td>1 GHz</td>
<td>1.25 GHz</td>
</tr>
<tr>
<td>Type</td>
<td>in-order</td>
<td>out-of-order</td>
<td>in-order</td>
</tr>
<tr>
<td>Issue Width</td>
<td>1</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>Instruction Window Size</td>
<td>-</td>
<td>64</td>
<td>-</td>
</tr>
<tr>
<td>Cache Line Size</td>
<td>64 bytes</td>
<td>64 bytes</td>
<td>64 bytes</td>
</tr>
<tr>
<td>L1 Cache Size</td>
<td>64 KB</td>
<td>64 KB</td>
<td>64 KB</td>
</tr>
<tr>
<td>L1 Cache Associativity</td>
<td>2-way</td>
<td>2-way</td>
<td>2-way</td>
</tr>
<tr>
<td>L2 Cache Size</td>
<td>1 MB</td>
<td>1.5 MB</td>
<td>1.5 MB</td>
</tr>
<tr>
<td>L2 Cache Associativity</td>
<td>8-way</td>
<td>6-way</td>
<td>6-way</td>
</tr>
<tr>
<td>L1 Hit / L2 Miss Latency</td>
<td>16 ns / 24 ns</td>
<td>12 ns / NA</td>
<td>12 ns / 16 ns</td>
</tr>
<tr>
<td>Local Memory Latency</td>
<td>80 ns</td>
<td>80 ns</td>
<td>80 ns</td>
</tr>
<tr>
<td>Remote Memory Latency</td>
<td>120 ns</td>
<td>120 ns</td>
<td>120 ns</td>
</tr>
<tr>
<td>Remote Data Latency</td>
<td>180 ns</td>
<td>180 ns</td>
<td>180 ns</td>
</tr>
</tbody>
</table>

Table 1: Parameters for different processor designs.
**Piranha Performance**

- **OLTP**
  - OOO 2.3x P1 (single processor piranha)
  - 1.6x due to aggressive technology
  - 1.45x due to wide OOO
  - P8 is 3x OOO, however

![OLTP Diagram](image)

**OLTP, contd.**

- **Near linear speedup**
  - High thread level parallelism
  - Highly effective on-chip caches

![Speedup Graph](image)
OLTP, contd.

- Cache effectiveness
  - L2 hits drop from .9 to .4
  - L2 fwd to L1 takes up slack
  - non-inclusive caches increase effective on-chip cache size
  - OLTP workloads have constructive cache interference

Piranha Performance

- DSS
  - P8 is 2.3x wide OOO
  - DSS has better spatial locality in data and better temporal locality in instructions
    => fewer memory stalls
    » helps OOO relatively more
Full Custom Piranha

- OLTP
  - 5x OOO
- DSS
  - 5.3x OOO
  - DSS is processor bound
  - more aggressive technology helps Piranha

![Normalized Execution Time Graph]

(c) 2003 J. E. Smith  CS/ECE 757