ECE 1749H: Interconnection Networks for Parallel Computer Architectures:

Interface with System Architecture

Prof. Natalie Enright Jerger

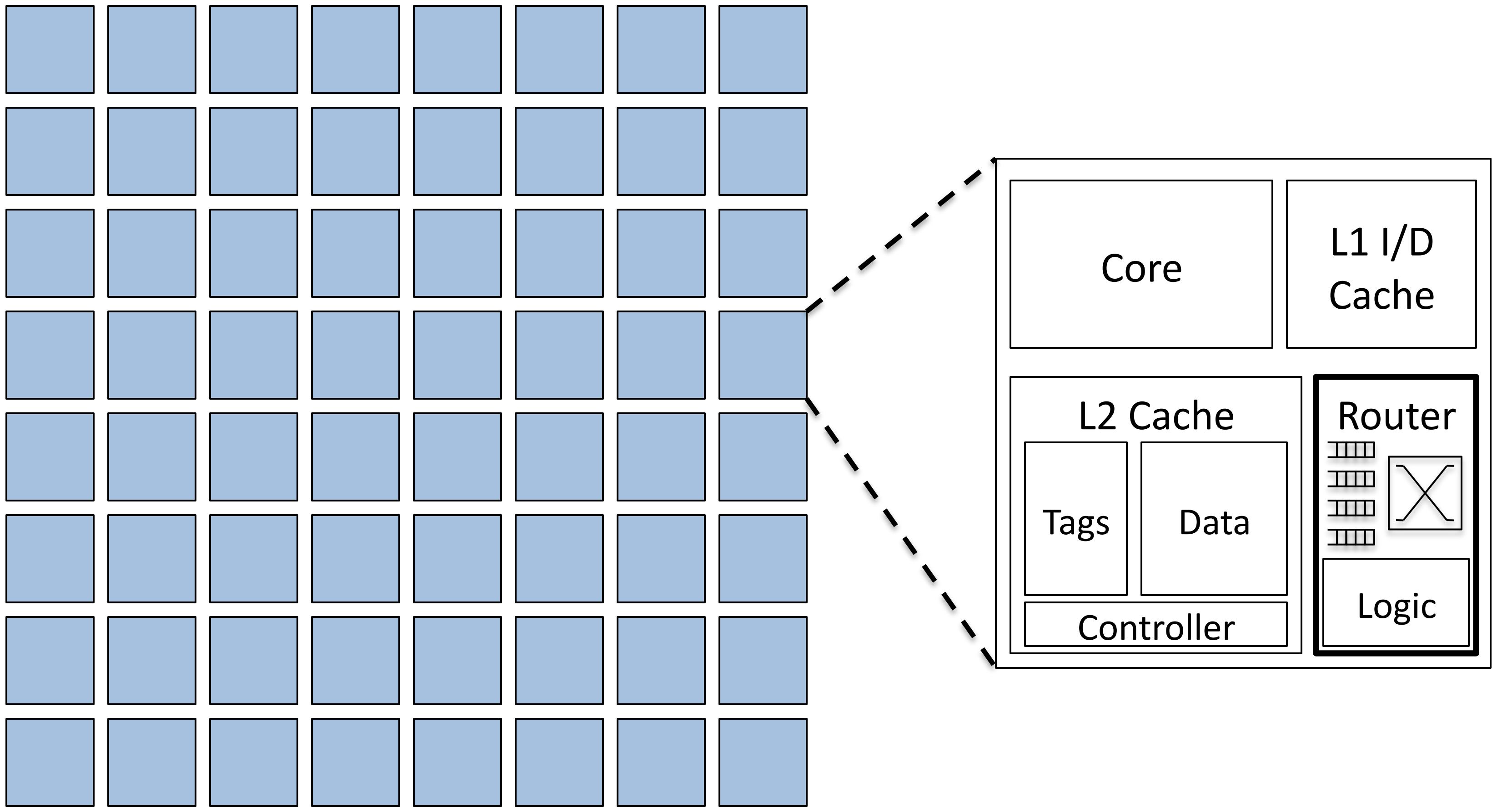
Systems and Interfaces

- Look at how systems interact and interface with network
- Two systems
  - Shared-memory chip multiprocessors
    - From high end servers to embedded products
  - Multiprocessor System on Chip (MPSoC)
    - Mobile consumer market
- Many more systems/applications for interconnects

Memory Model in CMPs

- Message Passing
  - **Explicit** movement of data between nodes and address spaces
  - Programmers manage communication
- Shared Memory
  - Communication occurs **implicitly** through loads/stores and accessing instructions
- Will focus on shared memory

Shared Memory CMP Architecture



Shared Memory Network for CMPs

- Logically…
  - all processors access same shared memory
- Practically…
  - cache hierarchies reduce access latency to improve performance
- Requires cache coherence protocol
  - to maintain **coherent** view in presence of **multiple** shared copies

Impact of Coherence Protocol on Network Performance

- Coherence protocol shapes communication needed by system
- Single writer, multiple reader invariant
- Requires:
  - Data requests
  - Data responses
  - Coherence permissions

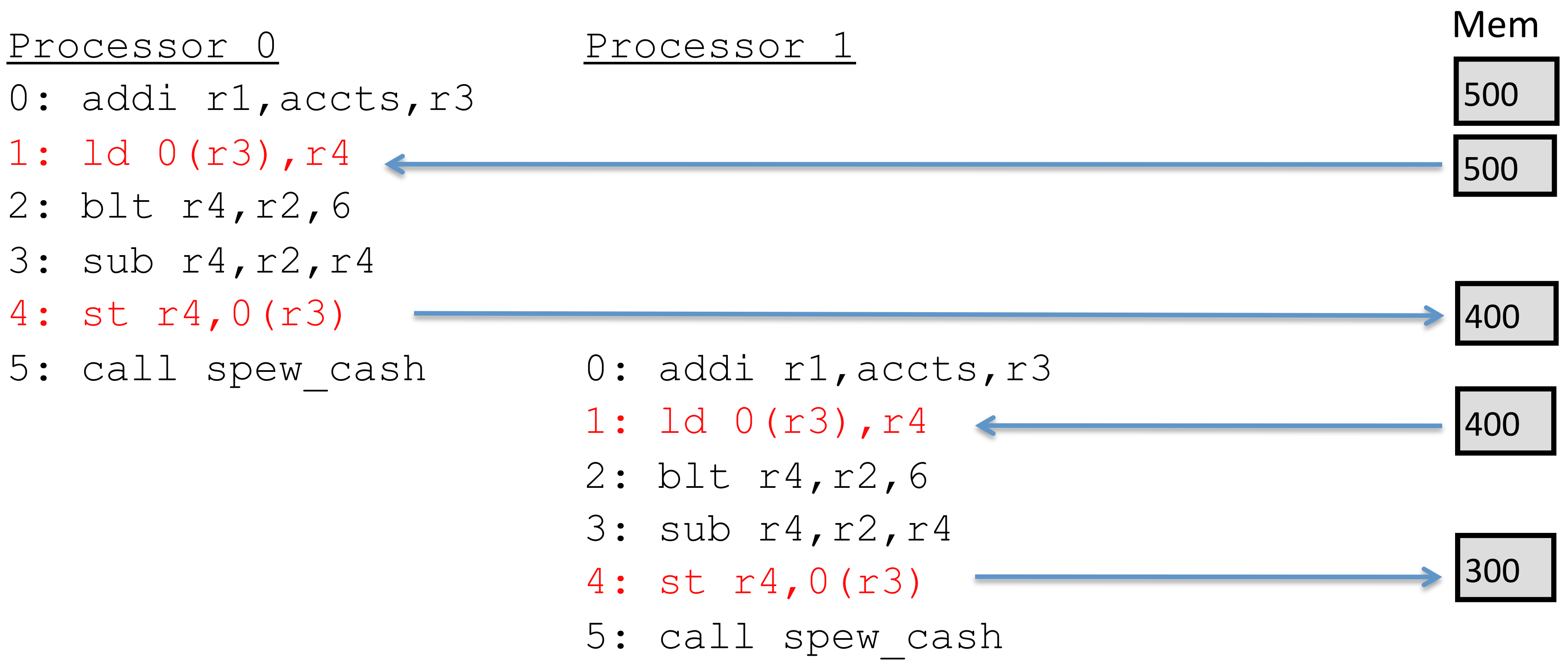
An Example Execution

```
Processor 0              Processor 1
0: addi r1,accts,r3
1: ld 0(r3),r4
2: blt r4,r2,6
3: sub r4,r2,r4
4: st r4,0(r3)
5: call spew_cash        0: addi r1,accts,r3
                         1: ld 0(r3),r4
                         2: blt r4,r2,6
                         3: sub r4,r2,r4
                         4: st r4,0(r3)
                         5: call spew_cash
```

CPU0	CPU1	Mem

- Two \$100 withdrawals from account #241 at two ATMs
  - Each transaction maps to thread on different processor
  - Track `accts[241].bal` (address is in `r3`)

No-Cache, No-Problem

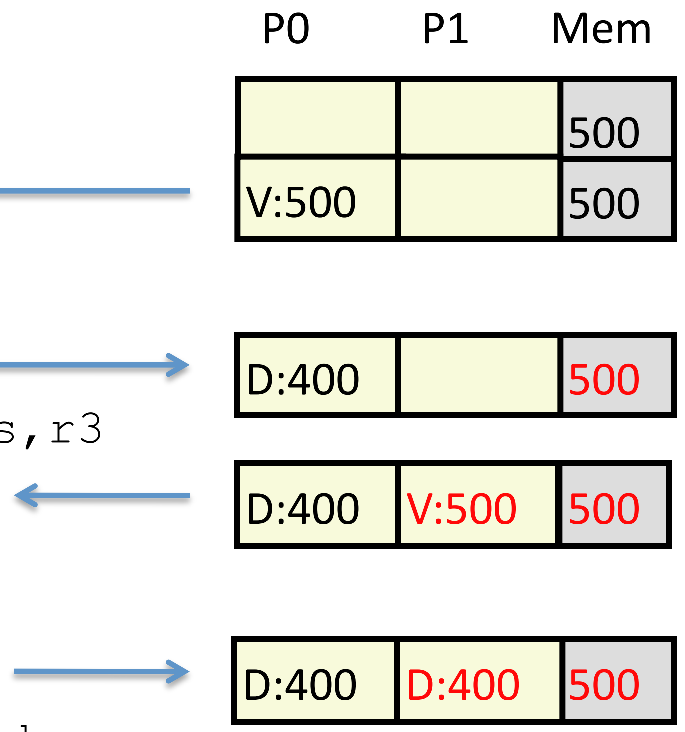


```
Processor 0              Processor 1
0: addi r1,accts,r3
1: ld 0(r3),r4
2: blt r4,r2,6
3: sub r4,r2,r4
4: st r4,0(r3)
5: call spew_cash        0: addi r1,accts,r3
                         1: ld 0(r3),r4
                         2: blt r4,r2,6
                         3: sub r4,r2,r4
                         4: st r4,0(r3)
                         5: call spew_cash
```

- Scenario I: processors have no caches
  - No problem

Cache Incoherence

```
Processor 0                     Processor 1
0: addi r1,accts,r3
1: ld 0(r3),r4
2: blt r4,r2,6
3: sub r4,r2,r4
4: st r4,0(r3)
5: call spew_cash               0: addi r1,accts,r3
                                1: ld 0(r3),r4
                                2: blt r4,r2,6
                                3: sub r4,r2,r4
                                4: st r4,0(r3)
                                5: call spew_cash
```



- Scenario II: processors have write-back caches
  - Potentially 3 copies of `accts[241].bal`: memory, p0$, p1$
  - Can get incoherent (inconsistent)

What to Do?

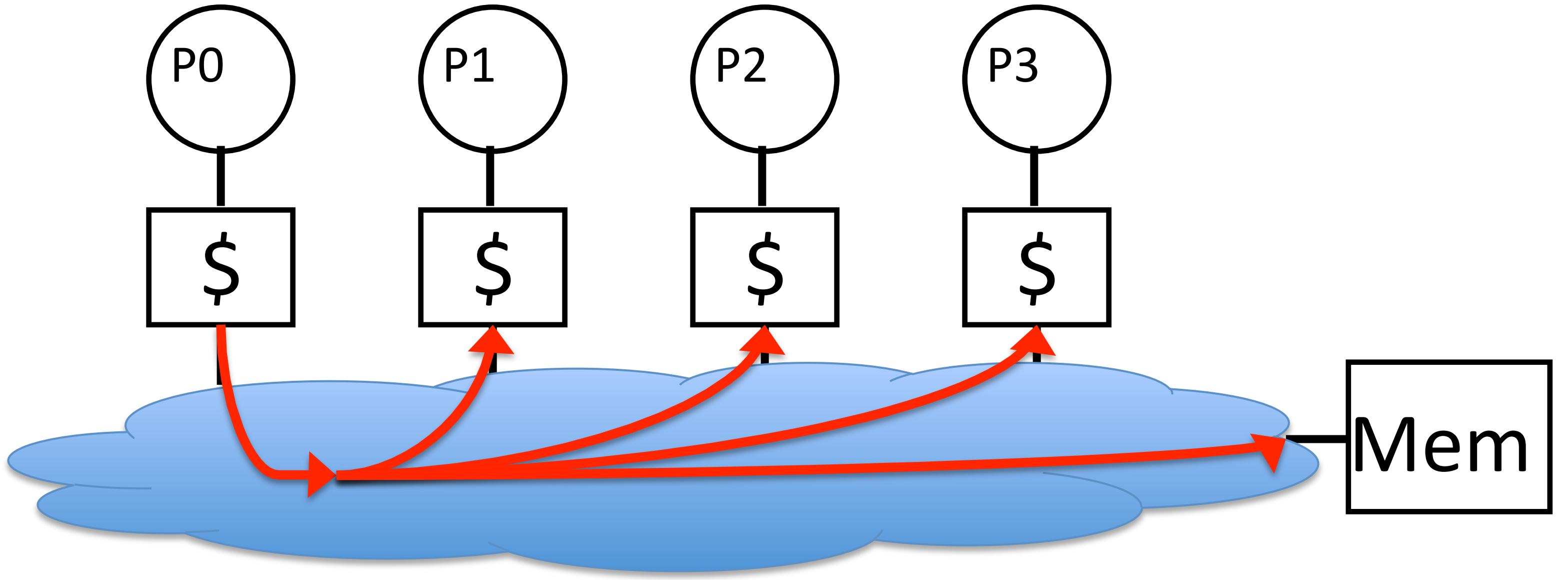
- No caches?
  - Slow!!
- Make shared data uncachable?
  - Faster, but still too slow
  - Entire accts database is technically "shared"
    - Definition of "loosely shared"
    - Data only really shared if two ATMs access same acct at once
  - Flush all other caches on writes to shared data?
    - May as well not have caches

Hardware Cache Coherence

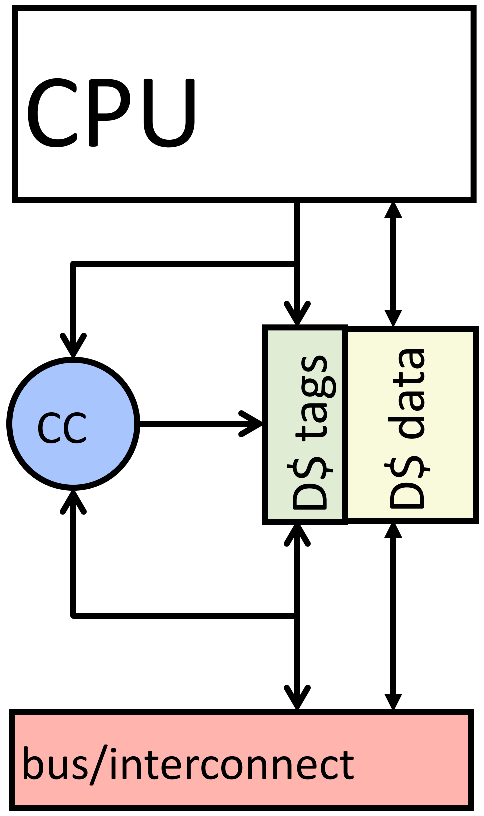
- Hardware cache coherence
  - Rough goal: all caches have same data at all times
  - Minimal flushing, maximum caches --> best performance

- Broadcast-based protocol
  - All processors see all requests at the same time, same order
  - Often rely on bus

Broadcast-based Coherence



Hardware Cache Coherence



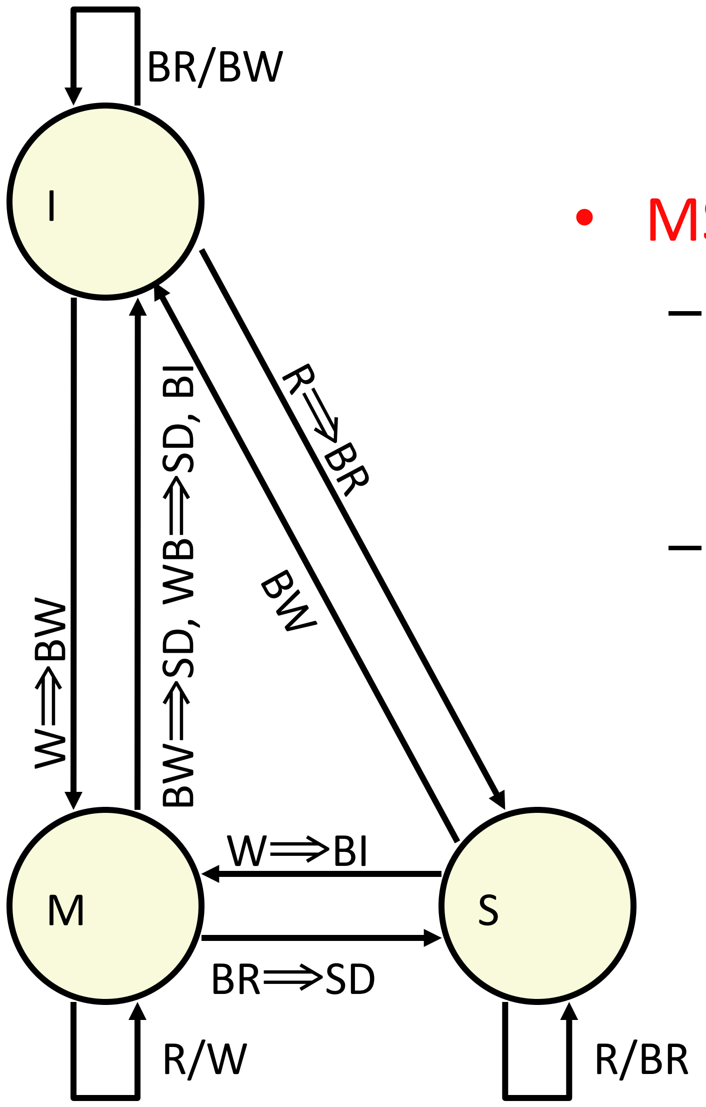
- Coherence
  - All copies have same data at all times
- Coherence controller:
  - Examines bus/interconnect traffic (addresses and data)
  - Executes coherence protocol
    - What to do with local copy when you see different things happening on bus

Coherence Events

- Cache actions
  - Three processor-initiated events
    - R: read
    - W: write
    - WB: write-back (select block for replacement)
  - Two bus-side events
    - BR: bus-read, read miss on another processor
    - BW: bus-write, write miss on another processor
  - One response event:
    - SD: send data

- Point-to-point network protocols also exist
  - Typical solution is a directory protocol

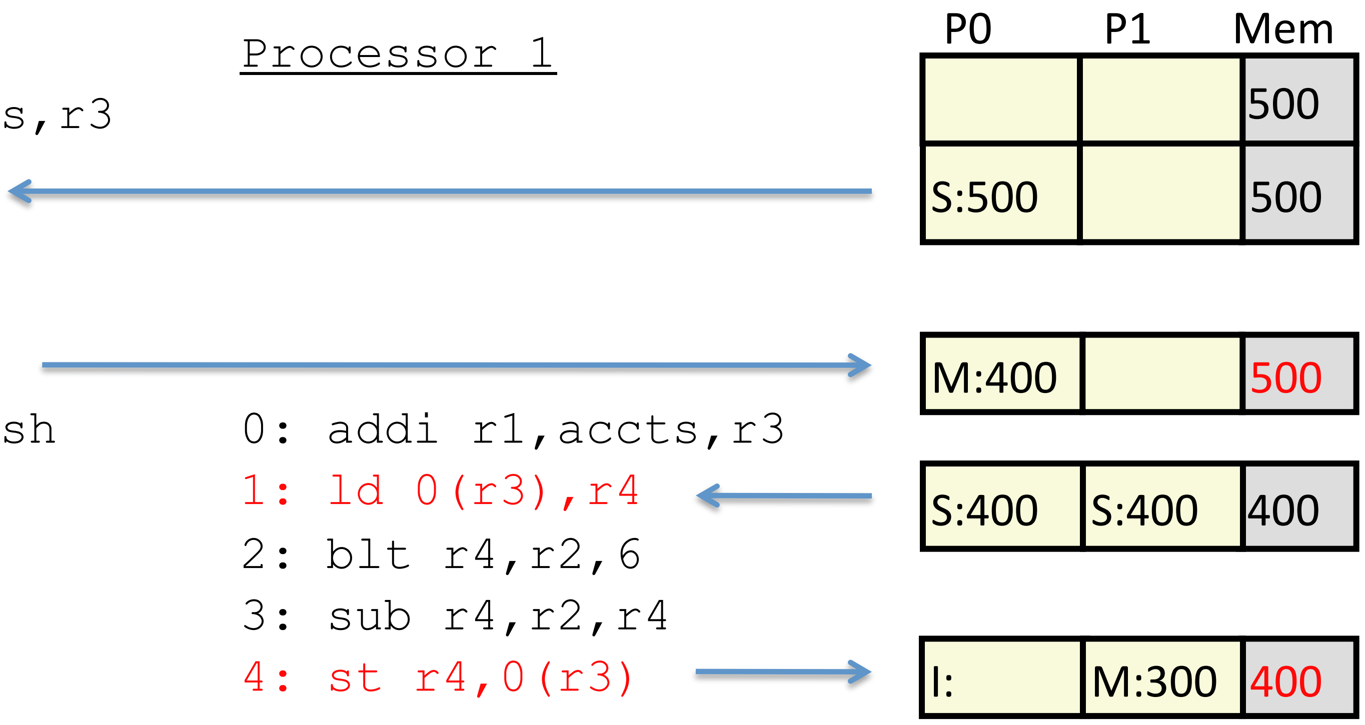
MSI Protocol



- MSI (modified-shared-invalid)
  - Two valid states
    - M (modified): local dirty copy
    - S (shared): local clean copy
  - Allows either
    - Multiple read-only copies (S-state) --OR--
    - Single read/write copy (M-state)

MSI Protocol (Write-Back Cache)

```
Processor 0              Processor 1
0: addi r1,accts,r3
1: ld 0(r3),r4
2: blt r4,r2,6
3: sub r4,r2,r4
4: st r4,0(r3)
5: call spew_cash        0: addi r1,accts,r3
                         1: ld 0(r3),r4
                         2: blt r4,r2,6
                         3: sub r4,r2,r4
                         4: st r4,0(r3)
                         5: call spew_cash
```



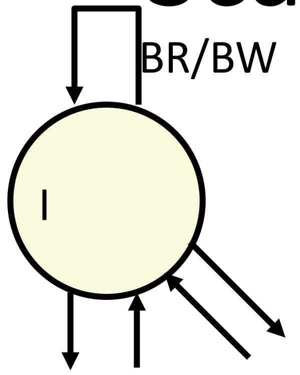
P0	P1	Mem
		500
S:500		500
M:400		500
S:400	S:400	400
I:	M:300	400

- `ld` by processor 1 generates a BR
  - processor 0 responds by Send Data its dirty copy, transitioning to S
- `st` by processor 1 generates a BW
  - processor 0 responds by transitioning to I

Coherence Bandwidth Requirements

- How much address bus bandwidth does snooping need?
  - Well, coherence events generated on…
    - Misses (only in L2, not so bad)
    - Dirty replacements
- Some parameters
  - 2 GHz CPUs, 2 IPC, 33% memory operations,
  - 2% of which miss in the L2, 50% of evictions are dirty
  - (0.33 * 0.02) + (0.33 * 0.02 * 0.50)) = 0.01 events/insn
  - 0.01 events/insn * 2 insn/cycle * 2 cycle/ns = 0.04 events/ns
  - Request: 0.04 events/ns * 4 B/event = 0.16 GB/s = 160 MB/s
  - Data Response: 0.04 events/ns * 64 B/event = 2.56 GB/s
- That's 2.5 GB/s … per processor
  - With 16 processors, that's 40 GB/s!
  - With 128 processors, that's 320 GB/s!!
  - Yes, you can use multiple buses… but that hinders global ordering

Scalable Cache Coherence

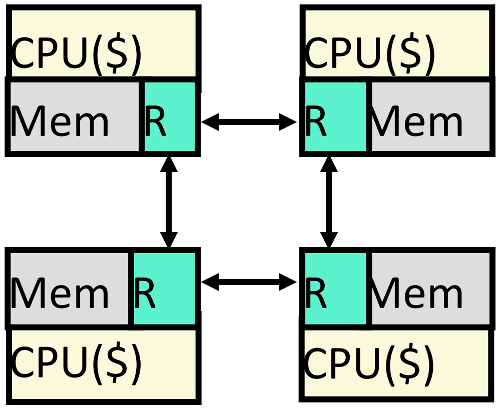


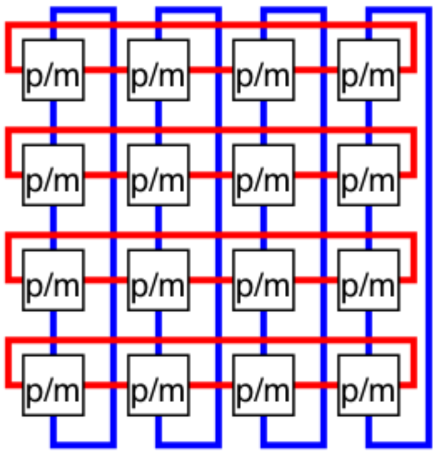
- Scalable cache coherence: two part solution

- Part I: bus bandwidth
  - Replace non-scalable bandwidth substrate (bus)…
  - …with scalable bandwidth substrate (point-to-point network, e.g., mesh)

- Part II: processor snooping bandwidth
  - Interesting: most snoops result in no action
  - Replace non-scalable broadcast protocol (spam everyone)…
  - …with scalable directory protocol (only spam processors that care)

Directory Coherence Protocols



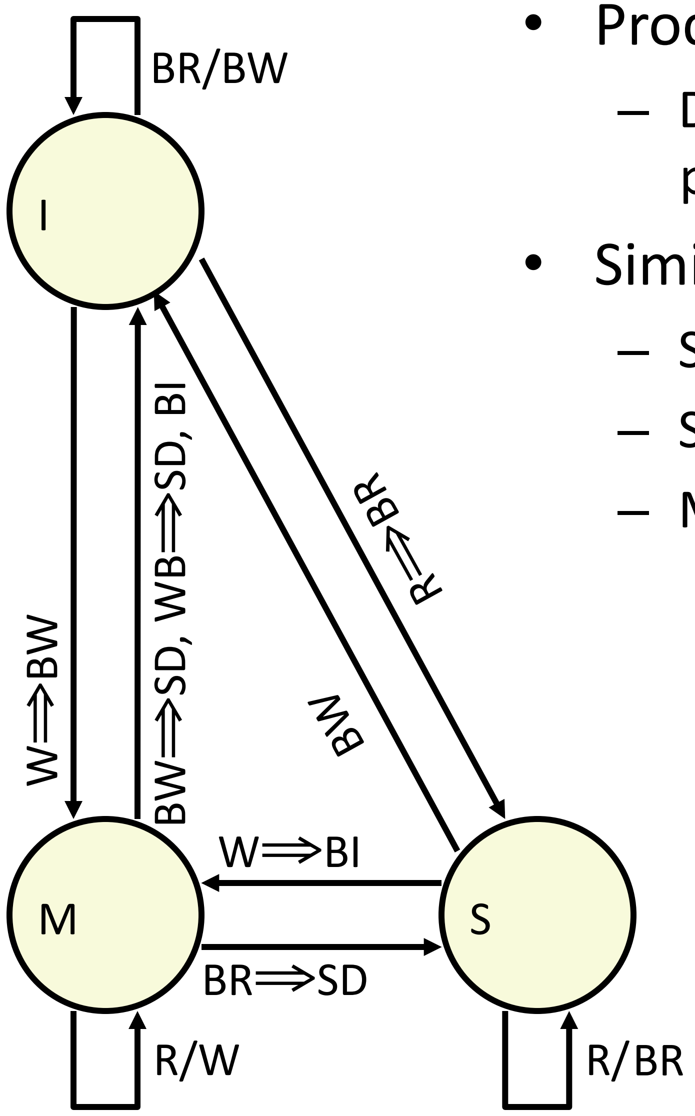


- Observe: physical address space statically partitioned (Still shared!!)
  + Can easily determine which memory module holds a given line
    - That memory module sometimes called "home"
  - Can't easily determine which processors have line in their caches
  – Bus-based protocol: broadcast events to all processors/caches
    ± Simple and fast, but non-scalable

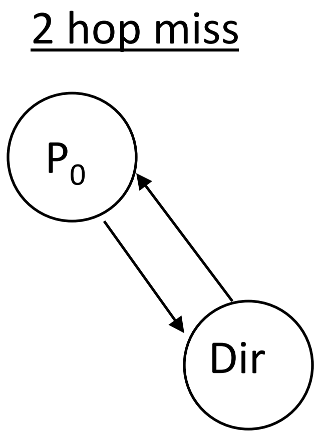
Directory Coherence Protocols

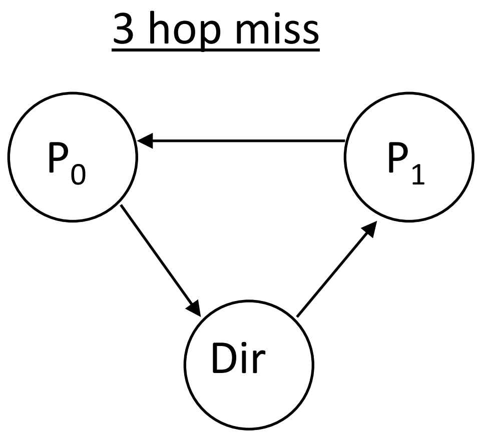
- Directories: non-broadcast coherence protocol
  - Extend memory to track caching information
  - For each physical cache line whose home this is, track:
    - Owner: which processor has a dirty copy (I.e., M state)
    - Sharers: which processors have clean copies (I.e., S state)
  - Processor sends coherence event to home directory
    - Home directory only sends events to processors that care

MSI Directory Protocol



- Processor side
  - Directory follows its own protocol (obvious in principle)
- Similar to bus-based MSI
  - Same three states
  - Same five actions (keep BR/BW names)
  - Minus grayed out arcs/actions
    - Bus events that would not trigger action anyway
    + Directory won't bother you unless you need to act

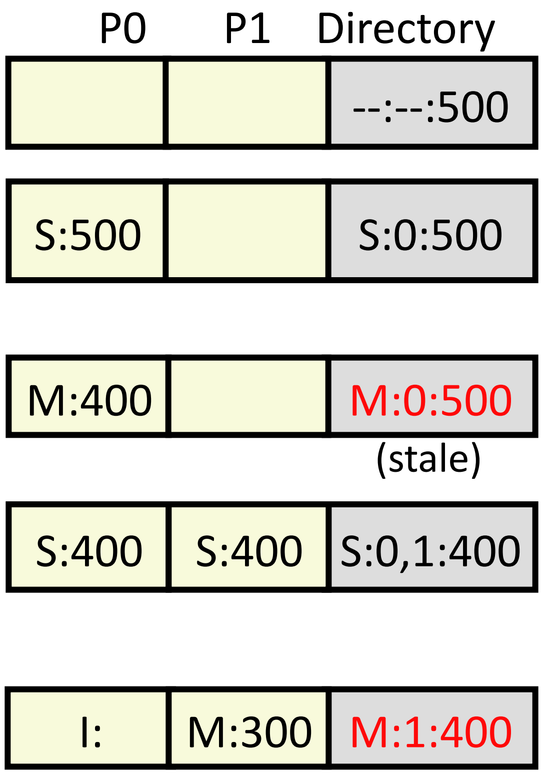




Directory MSI Protocol

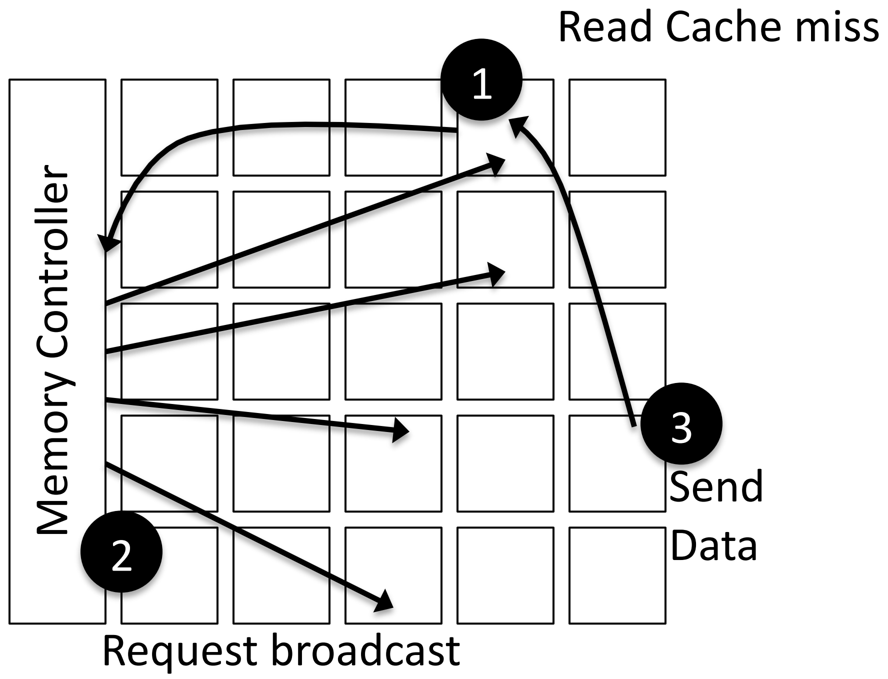
```
Processor 0
0: addi r1,accts,r3
1: ld 0(r3),r4
2: blt r4,r2,6
3: sub r4,r2,r4
4: st r4,0(r3)
5: call spew_cash
```

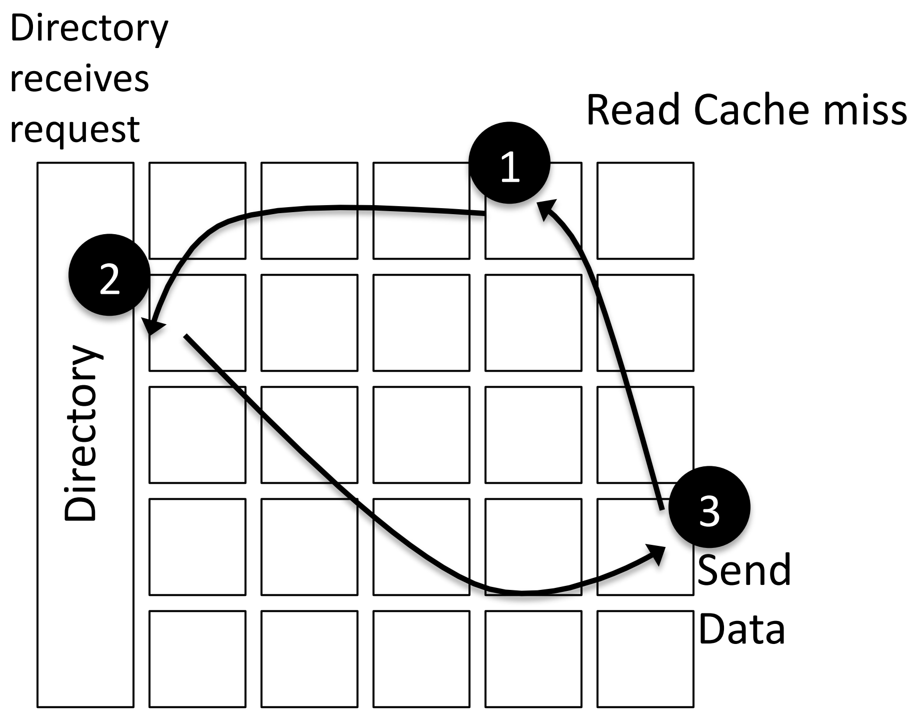
```
Processor 1
0: addi r1,accts,r3
1: ld 0(r3),r4
2: blt r4,r2,6
3: sub r4,r2,r4
4: st r4,0(r3)
5: call spew_cash
```



- ld by P1 sends BR to directory
  - Directory sends BR to P0, P0 sends P1 data, does WB, goes to S
- st by P1 sends BW to directory
  - Directory sends BW to P0, P0 goes to I

Broadcast vs. Directory

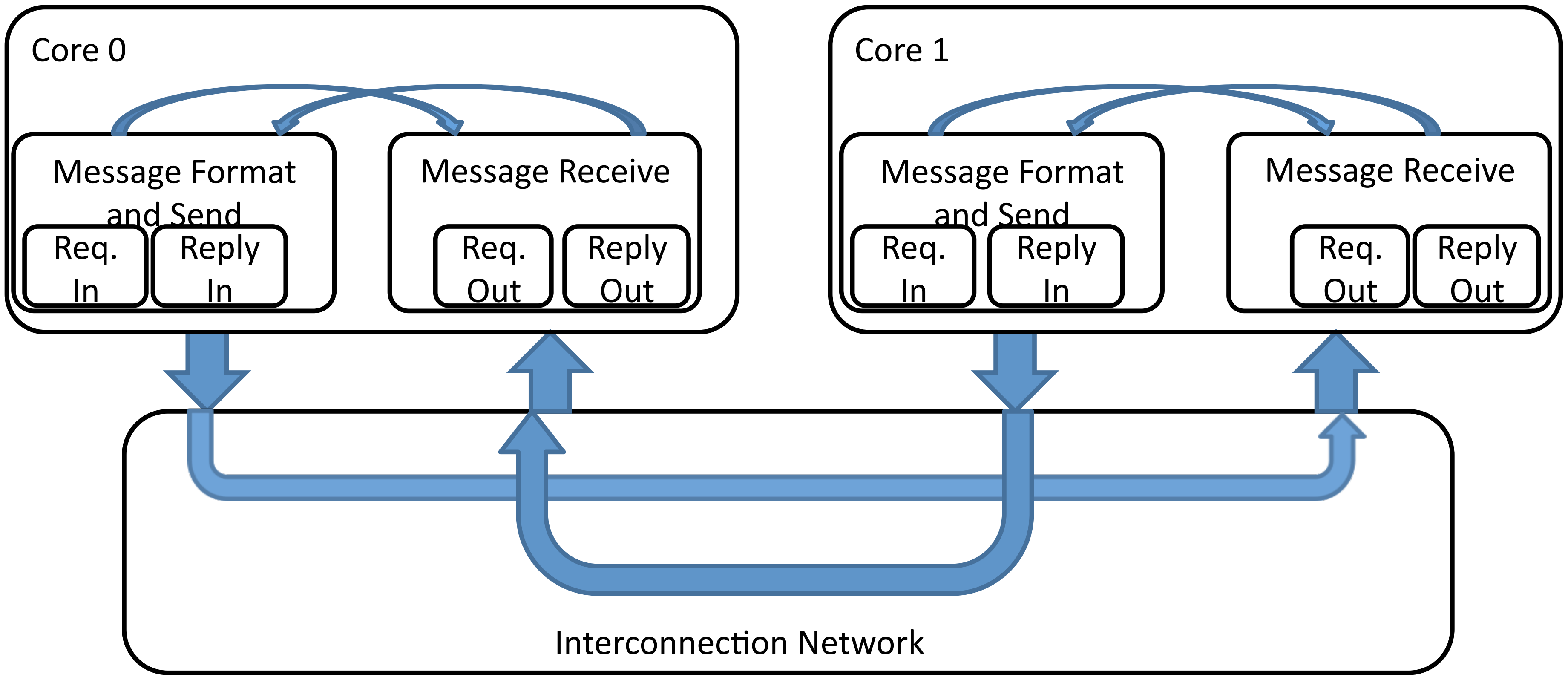




Coherence Protocol Requirements

- Different message types
  - Unicast, multicast, broadcast
- Directory protocol
  - Majority of requests: Unicast
    - **Lower bandwidth** demands on network
  - More scalable due to point-to-point communication
- Broadcast protocol
  - Majority of requests: Broadcast
    - **Higher bandwidth** demands
  - Often rely on **network ordering**

Protocol Level Deadlock



- Network becomes flooded with requests that cannot be consumed until the network interface has generated a reply
- Deadlock dependency between multiple message classes
- Virtual channels can prevent protocol level deadlock (to be discussed later)

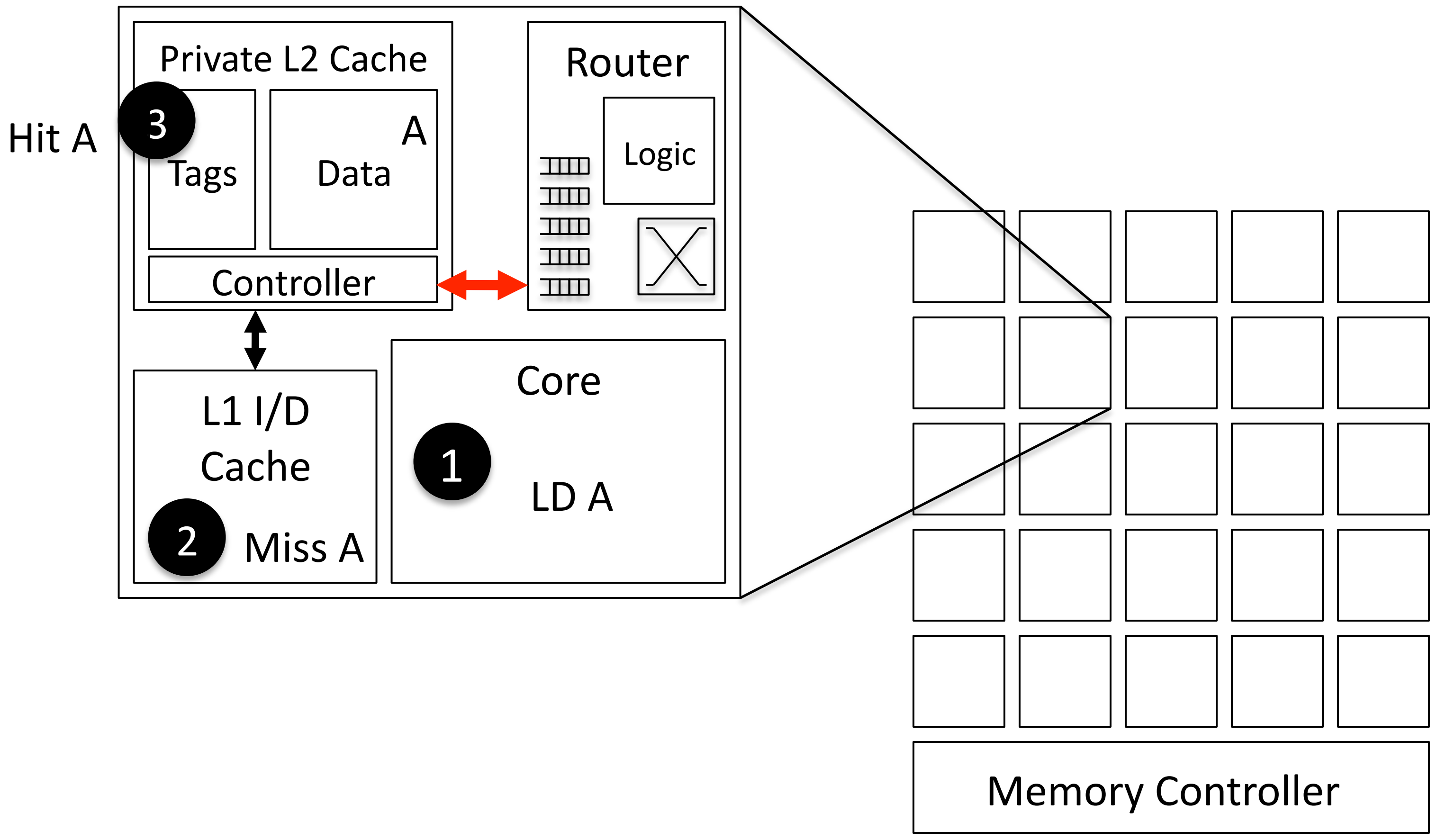
Impact of Cache Hierarchy

- Sharing of injection/ejection port among cores and caches

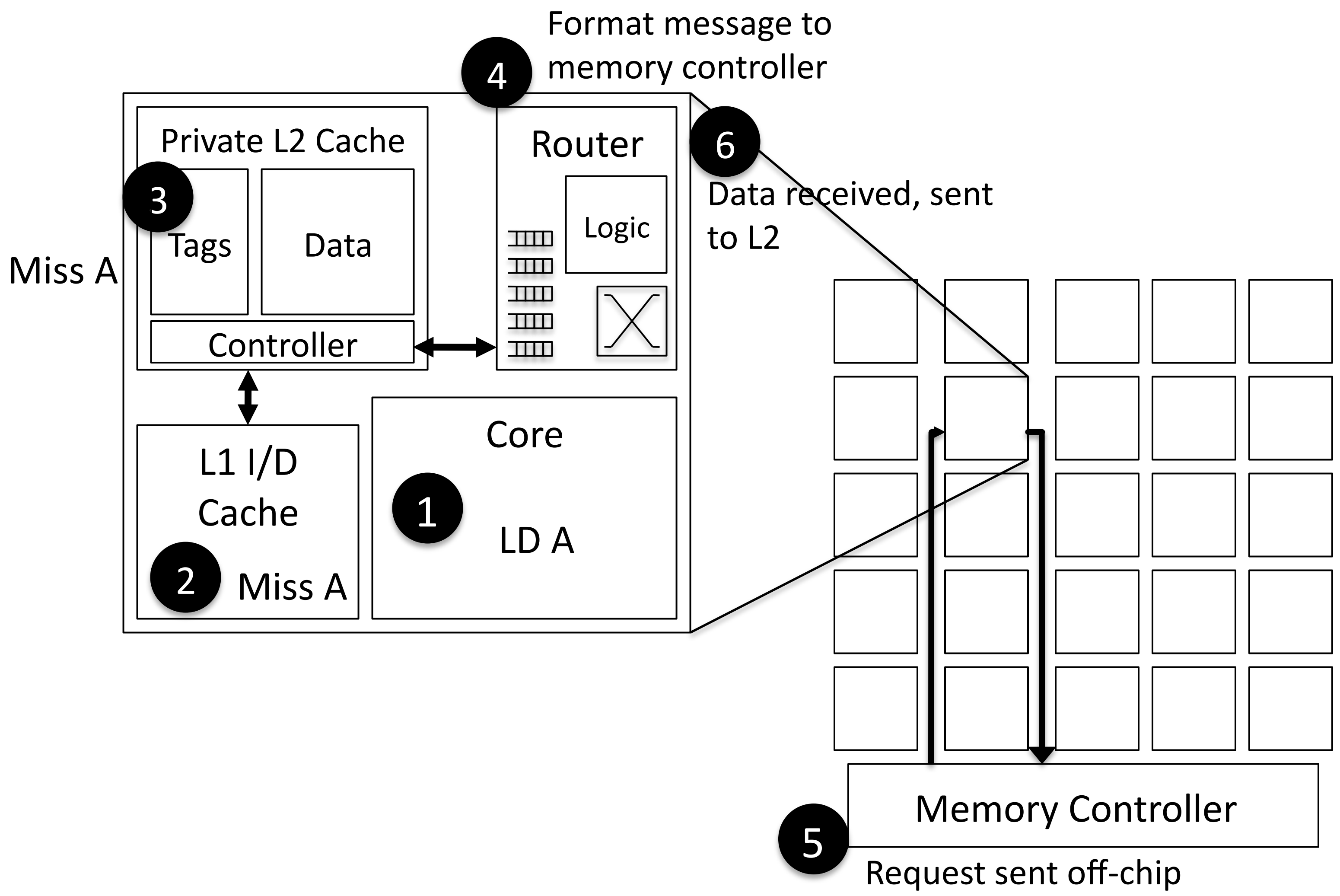
- Caches reduce average memory latency
  - Private caches
    - **Multiple** L2 copies
    - Data can be replicated to be close to processor
  - Shared caches
    - Data can only exist in **one** L2 to bank

- Serve as filter for interconnect traffic

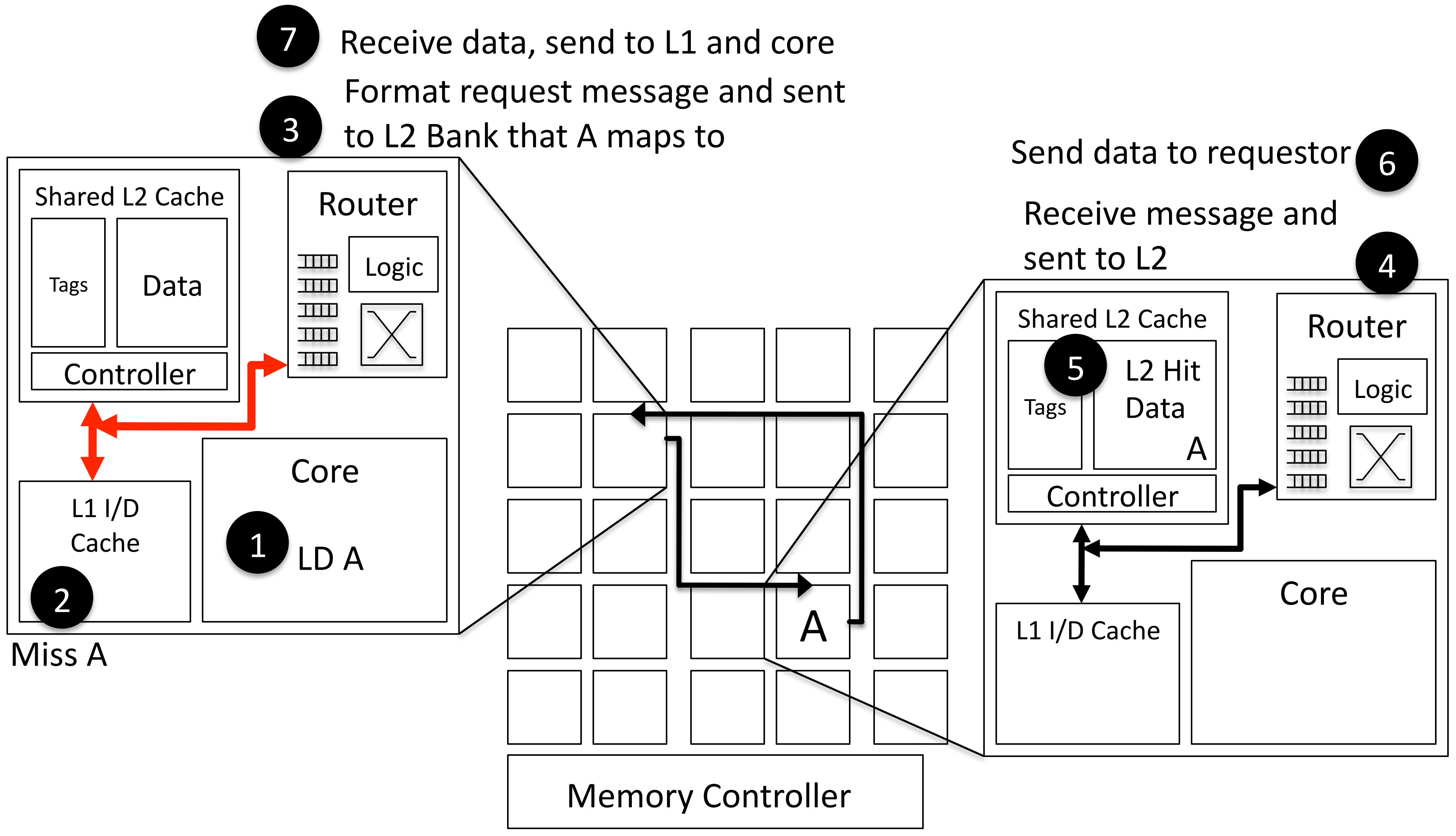
Private L2 Caches



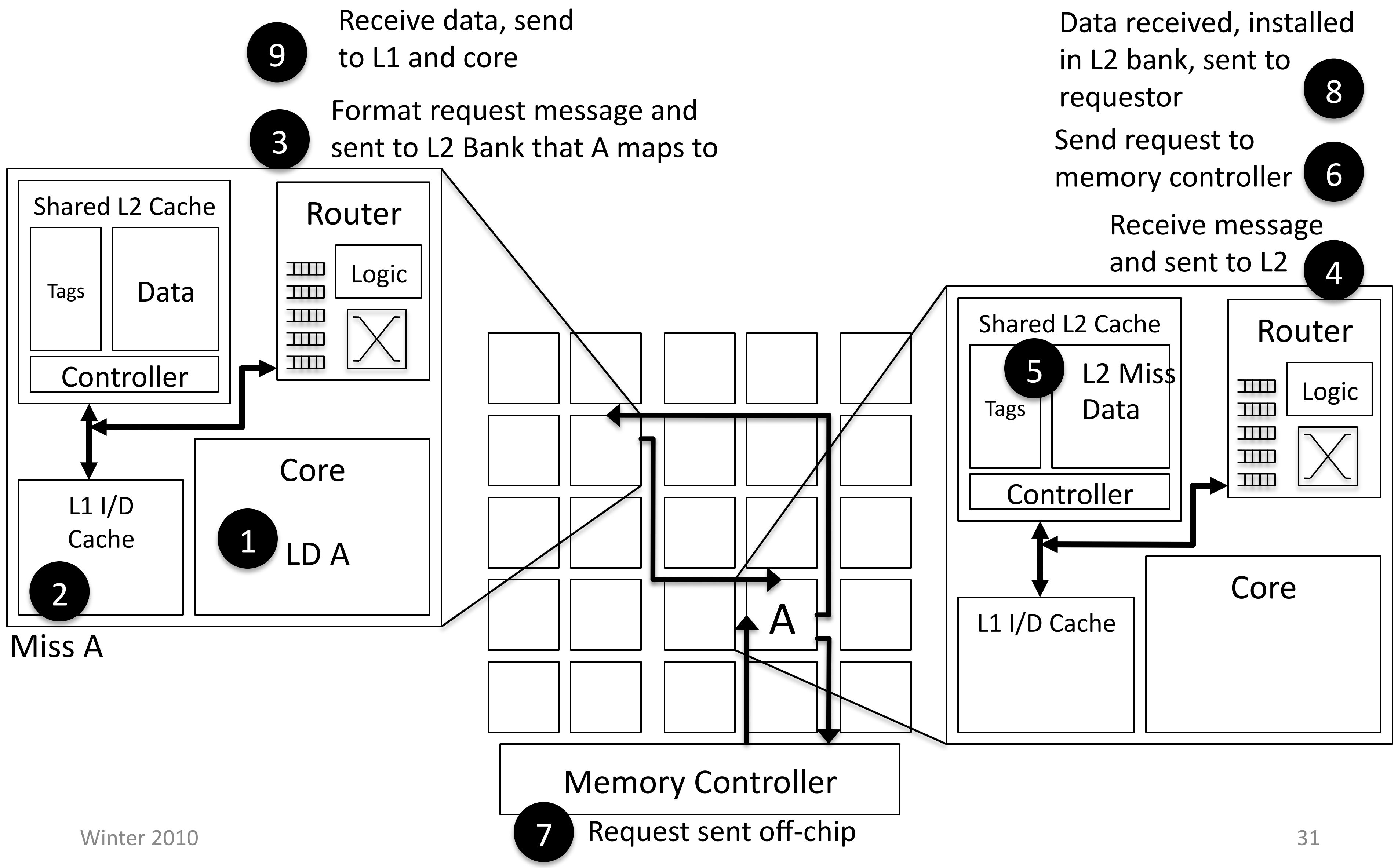
Private L2 Caches (2)



Shared L2 Caches



Shared L2 Caches (2)


9
Receive data, send to L1 and core
3
Format request message and sent to L2 Bank that A maps to
Data received, installed in L2 bank, sent to requestor
8
Send request to memory controller
6
Receive message and sent to L2
4
Shared L2 Cache
Tags
Data
Controller
Router
Logic
L1 I/D Cache
Core
1
LD A
2
Miss A
A
Shared L2 Cache
5
Tags
L2 Miss
Data
Controller
Router
Logic
L1 I/D Cache
Core
Memory Controller
7
Request sent off-chip

Private vs. Shared Caches

- Private caches
  - Reduce **latency** of L2 cache hits
    - keep frequently accessed data close to processor
  - Increase **off-chip** pressure

- Shared caches
  - Better use of storage
  - Non-uniform L2 hit latency
  - More on-chip network pressure
    - **all L1 misses** go onto network

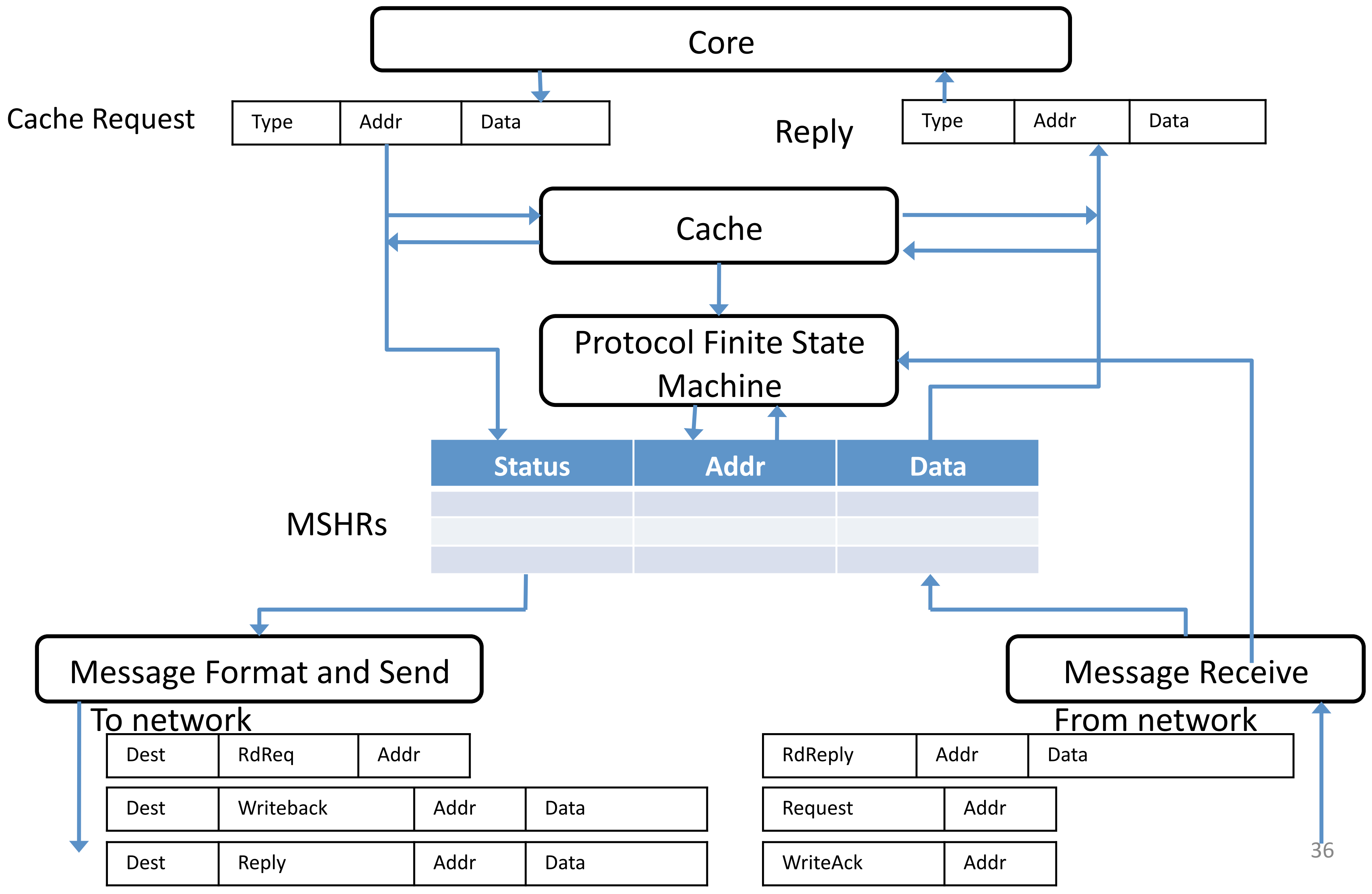
Home Node/Memory Controller Issues

- Heterogeneity in network
  - Some memory controller tiles
    - **Co-located** with processor/cache or **separate** tile
    - Share injection/ejection bandwidth?
- Home node
  - Directory coherence information
  - <= number of tiles
- Potential hot spots in network?

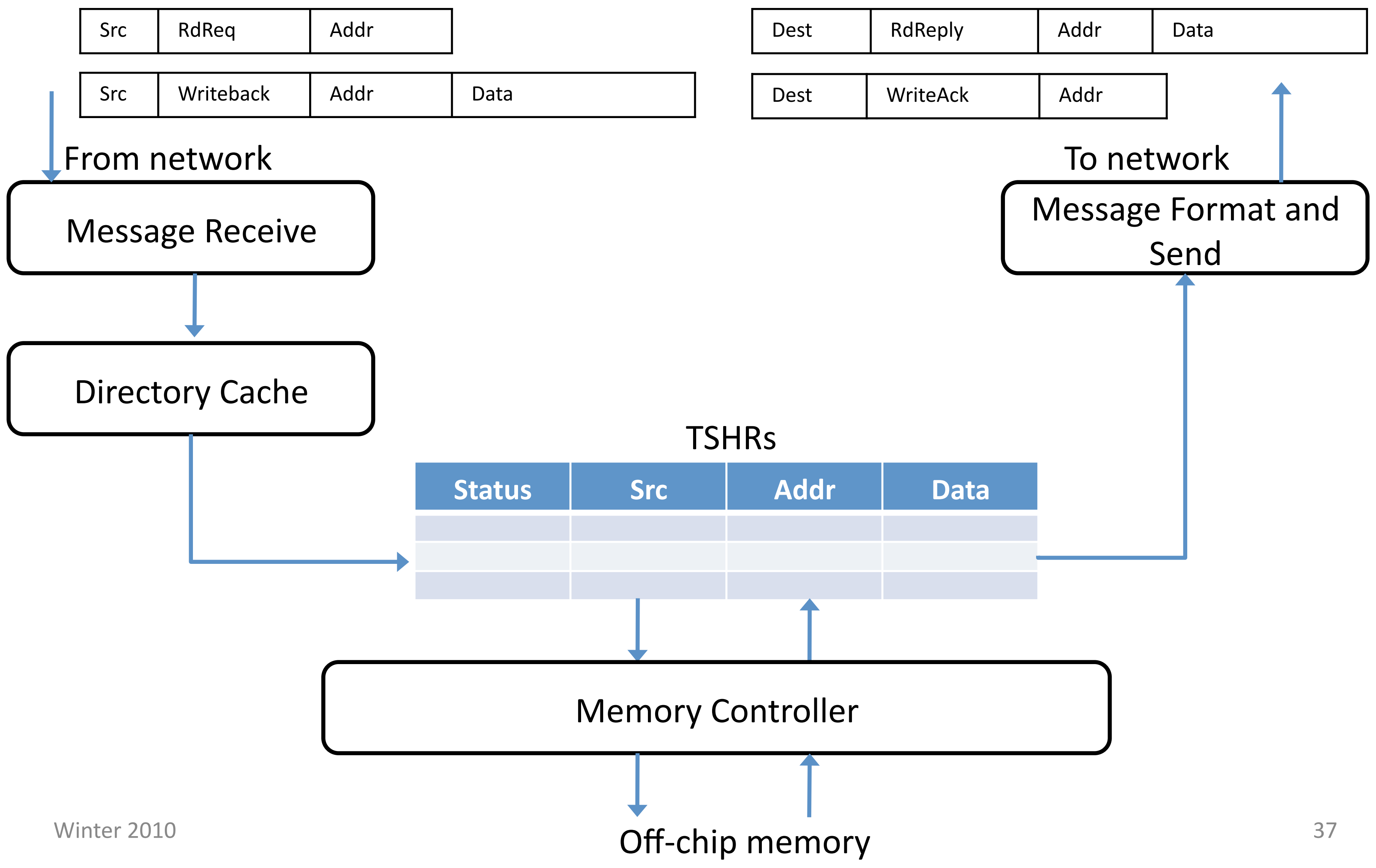
CMP Summary

- Cache hierarchies and coherence protocols
  - On-going areas of research for many-core
  - OCN cares about how various organizations impact traffic

Network Interface: Miss Status Handling Registers



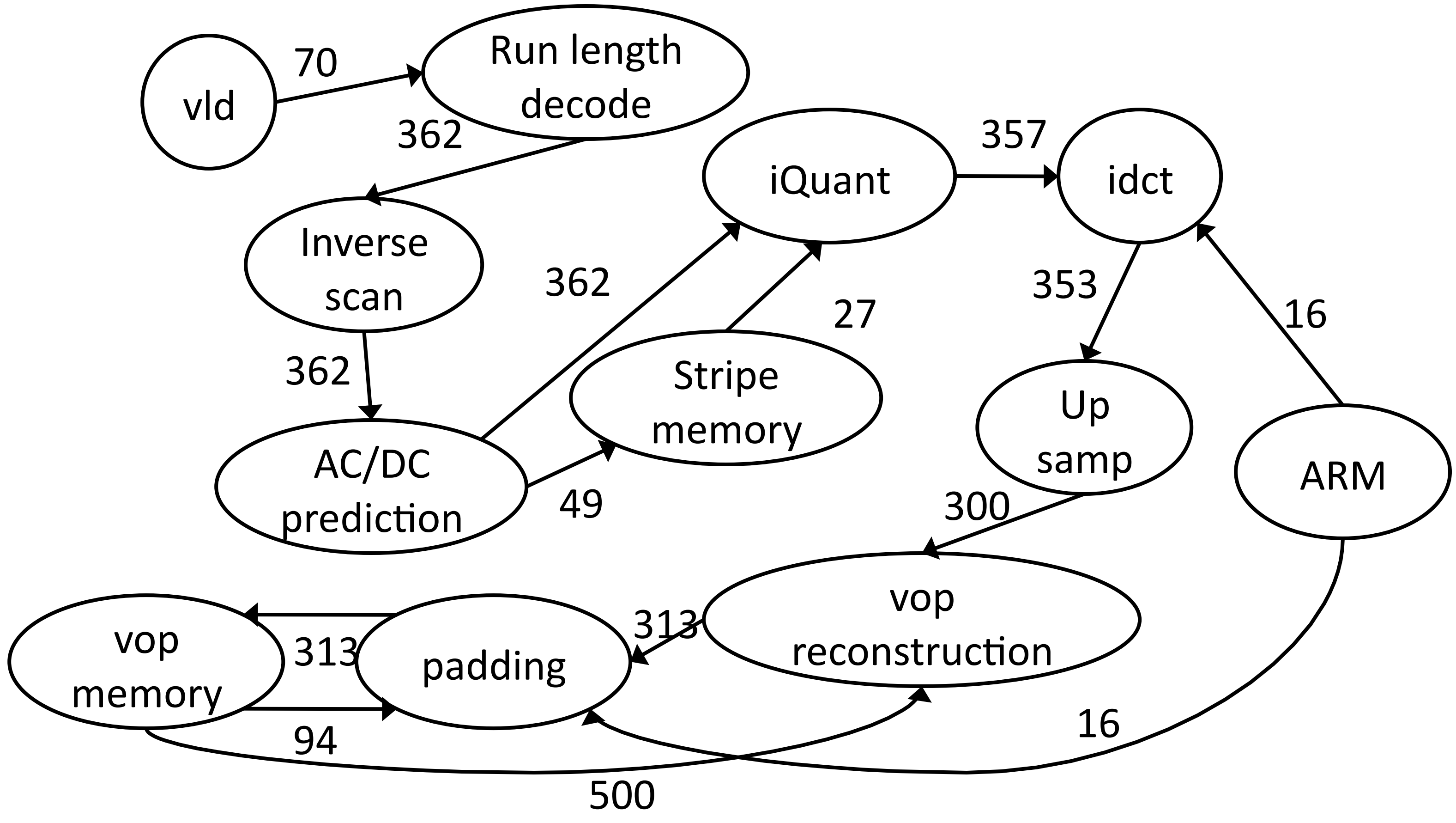
Transaction Status Handling Registers



Synthesized NoCs for MPSoCs

- System-on-Chip (SoC)
  - Chips tailored to specific applications or domains
  - Designed quickly through composition of IP blocks
- Fundamental NoC concepts applicable to both CMP and MPSoC
- Key characteristics
  - Applications known **a priori**
  - **Automated** design process
  - **Standardized** interfaces
  - Area/power constraints tighter

Application Characterization



- Describe application with task graphs
- Annotate with traffic volumes

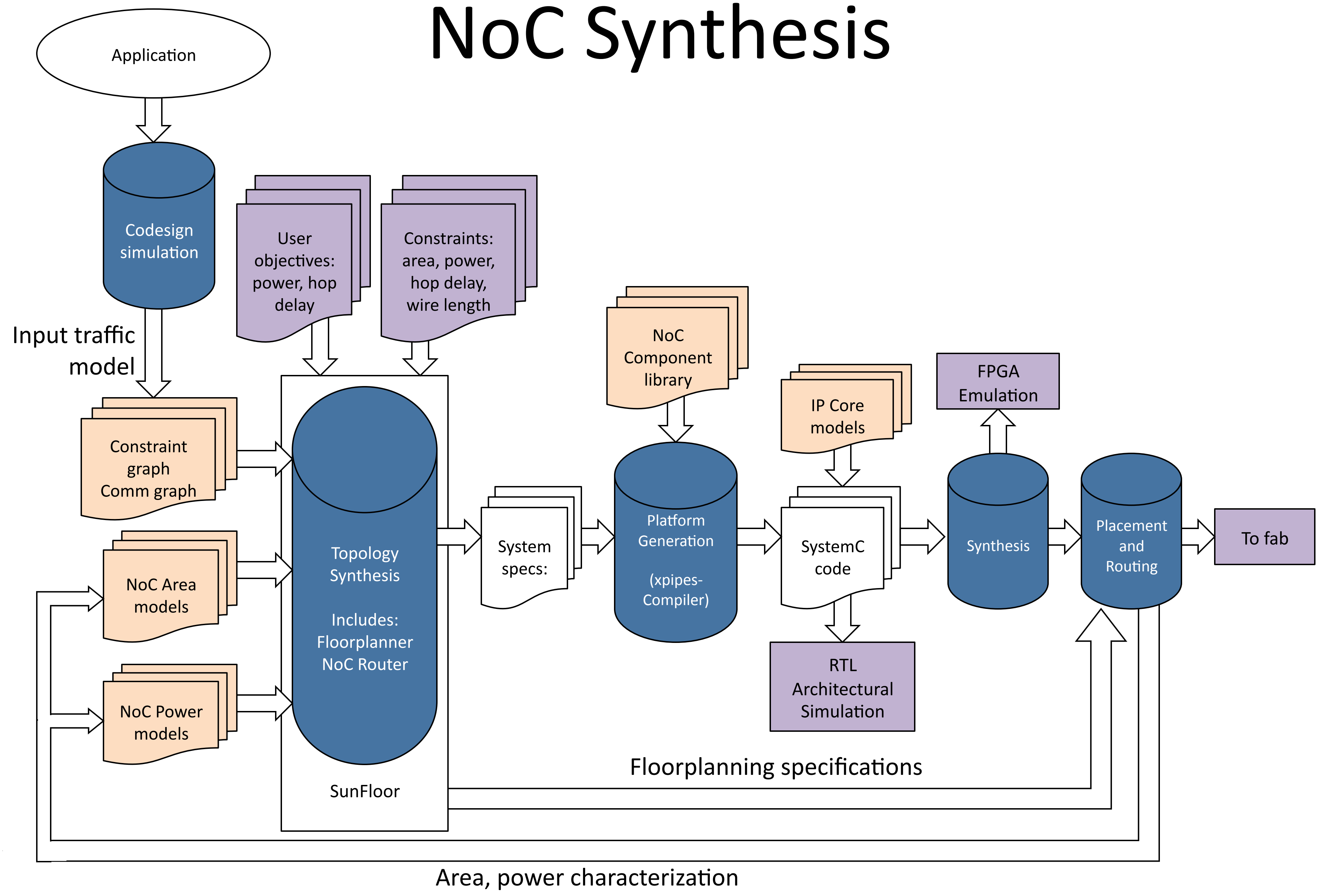
Design Requirements

- Less aggressive
  - CMPs: GHz clock frequencies
  - MPSoCs: MHz clock frequencies
  - Pipelining may not be necessary
  - Standardizes interfaces add significant delay

- Area and power
  - CMPs: 100W for server
  - MPSoC: several watts only

- Time to market
  - Automatic composition and generation

NoC Synthesis



NoC Synthesis

- Tool chain
  - Requires accurate power and area models
  - Quickly iterate through many designs
  - Library of soft macros for all NoC building blocks
  - Floorplanner
    - Determine router locations
    - Determine link lengths (delay)

NoC Network Interface Standards

- Standardized protocols
  - Plug and play with different IP blocks
- Bus-based semantics
  - Widely used
- Out of order transactions
  - Relax strict bus ordering semantics
  - Migrating MPSoCs from buses to NoCs.

Summary

- Architecture
  - Impacts communication requirements
  - Broadcast vs. Directory
  - Shared vs. Private Caches
- CMP vs. MPSoC
  - General vs. Application specific
  - Custom interfaces vs. standardized interfaces

Next Time

- Look at Topology and Routing

- Announcement:
  - Distinguished Lecture Tomorrow
  - Norm Jouppi, Director of the Exascale Computing Lab at HP
  - Talk: System Implications of Integrated Photonics