Cache Coherence Protocols

Avery Whitaker

COMP 522

01.22.2019

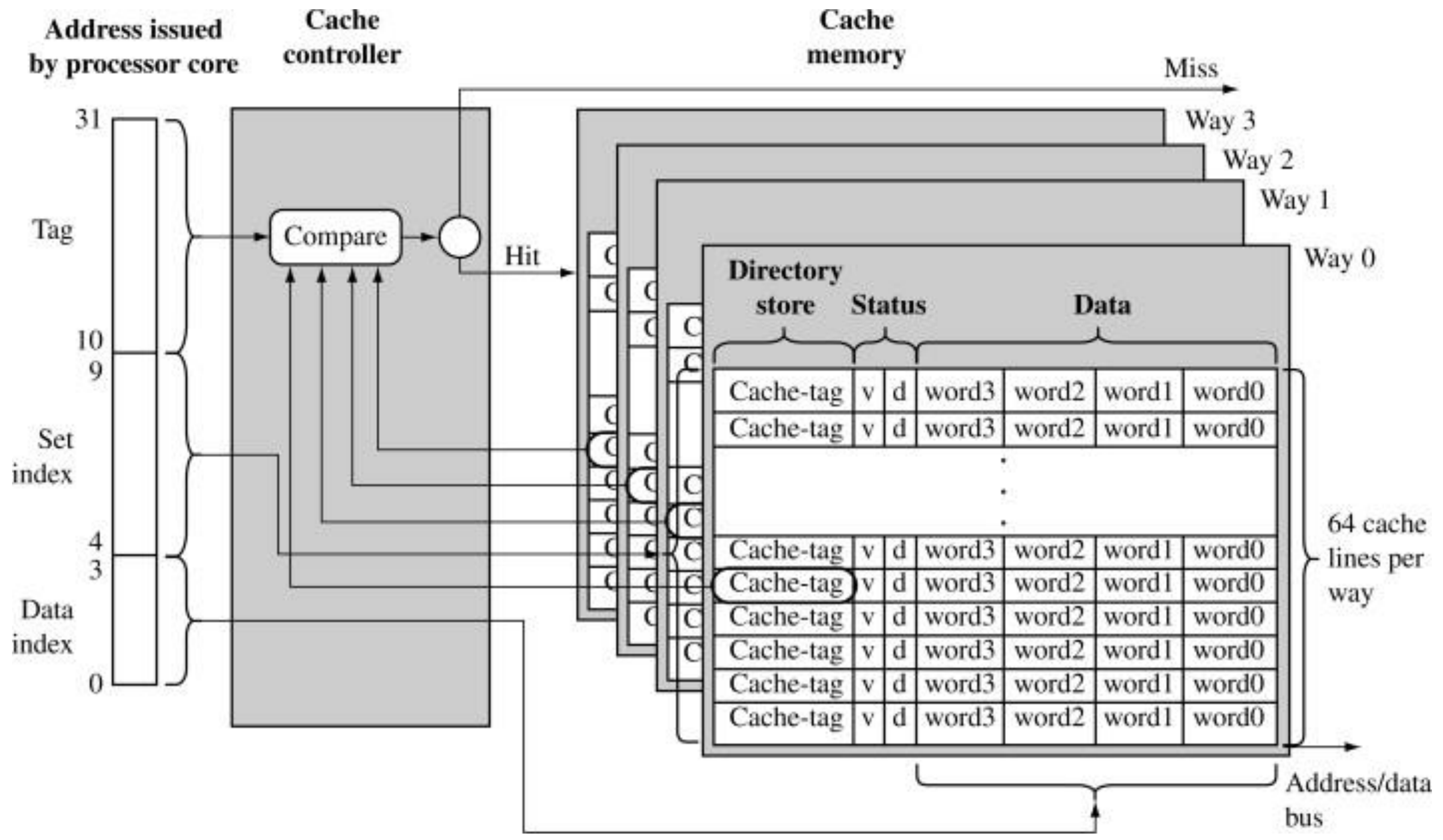
DATA
Multiprocessors

- ➢ **Caches**
- ➢ Consistency
- ➢ Coherence
- ➢ Victim Replication

Hiding Memory Latency

- How can we hide memory latency?
  - Prefetching
  - Out of order execution
  - Speculation
  - On-chip Cache

Set Associative Caches



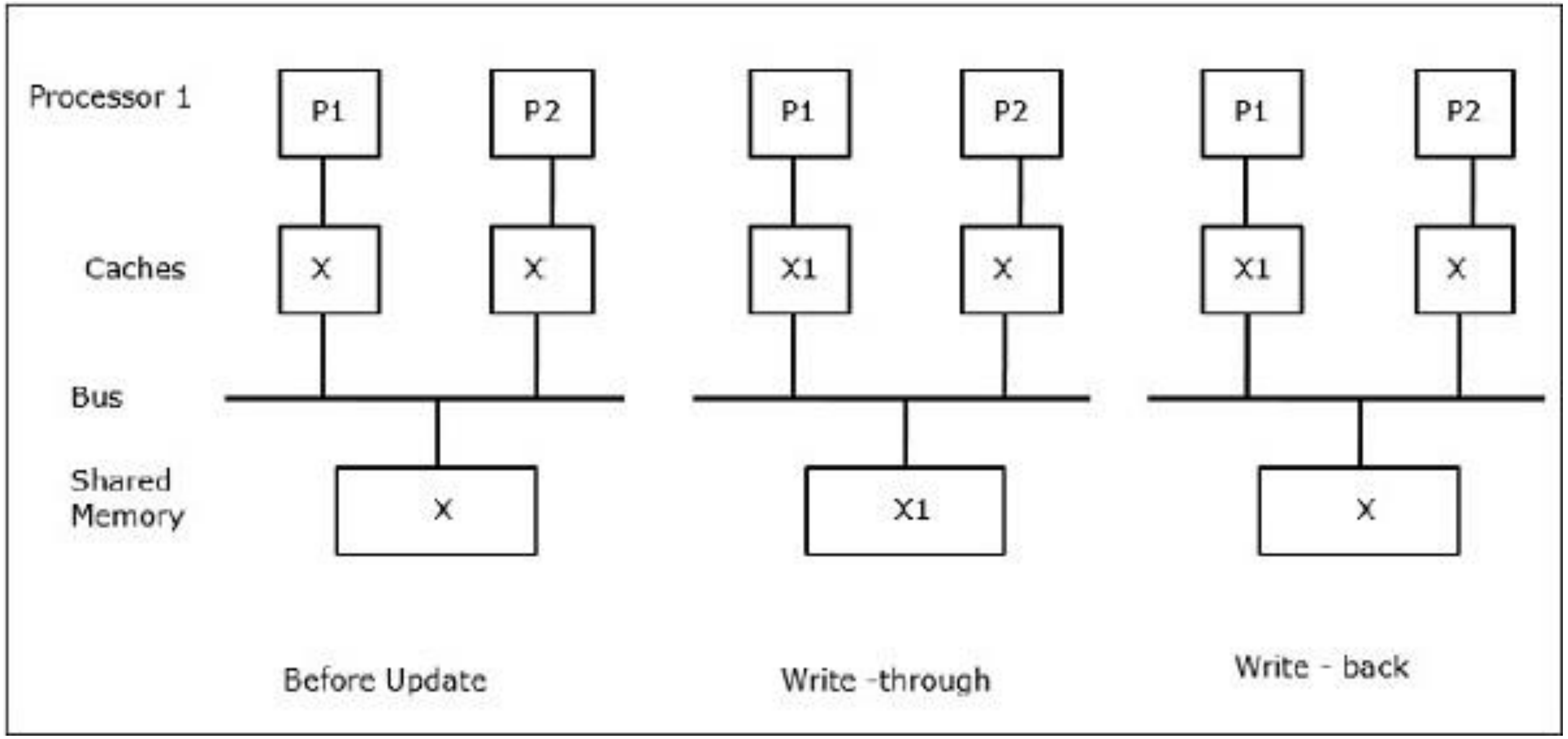
Andrew N. Sloss, Chris Wright, In Arm System Developer's Guide, 2004

Inclusive, Non-Inclusive, and Exclusive Caches

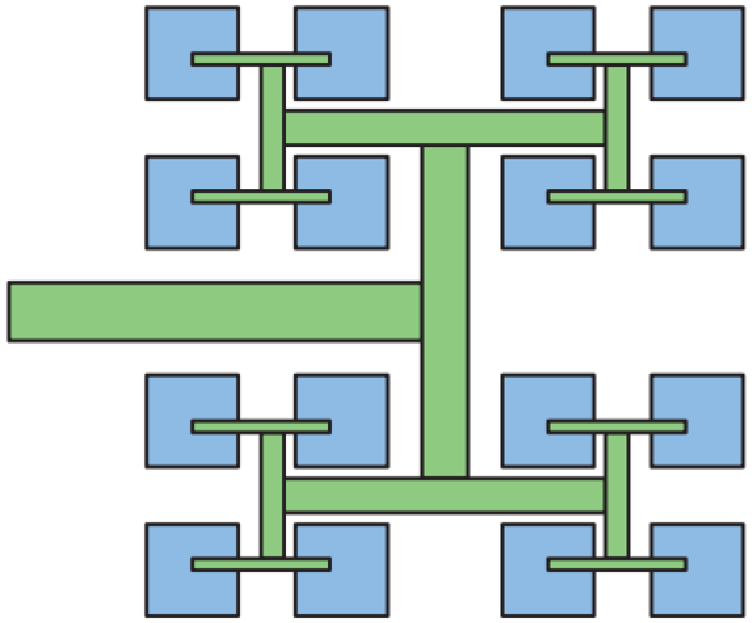
- If Data is in L1 cache, is it in L2 Cache?
  - Inclusive: Yes
  - Exclusive: No
  - Non-Inclusive: Maybe

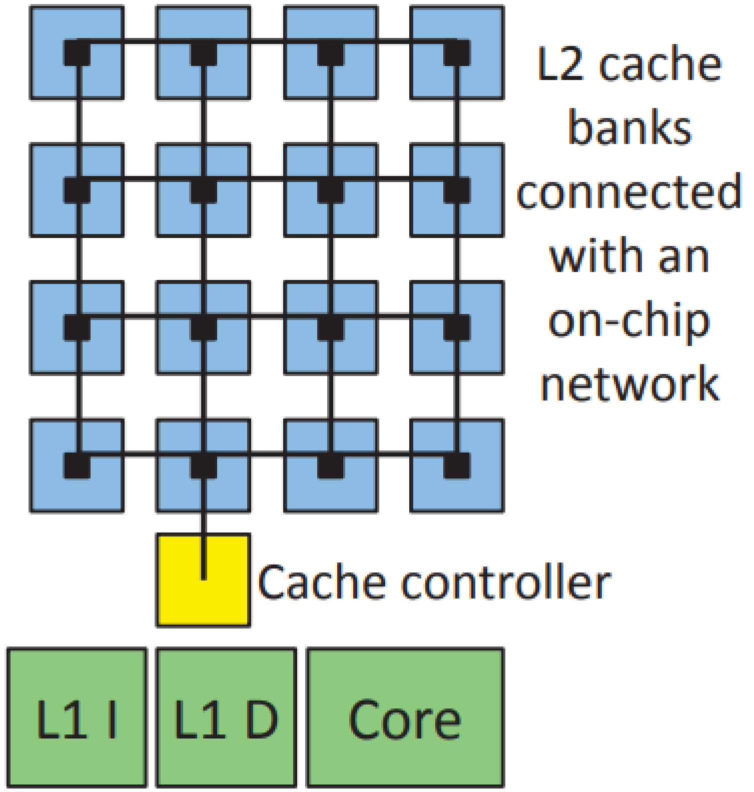
- Pros/Cons of each?
  - Need inclusion for other processors to get hit if multiple using same block
  - Inclusive duplicates data
  - More bandwidth then Non-Inclusive

Write-Back vs Write-Through

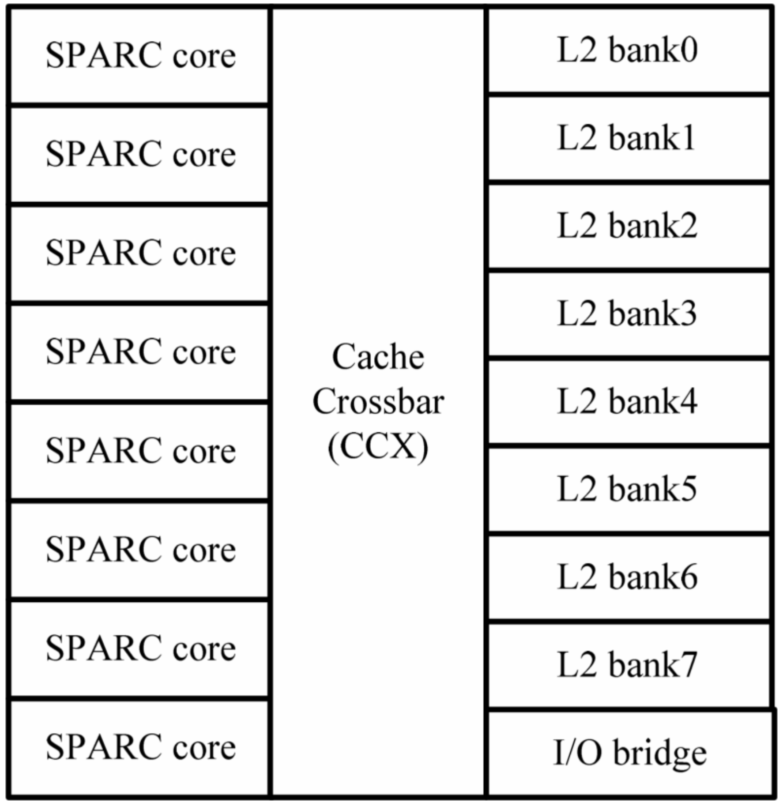


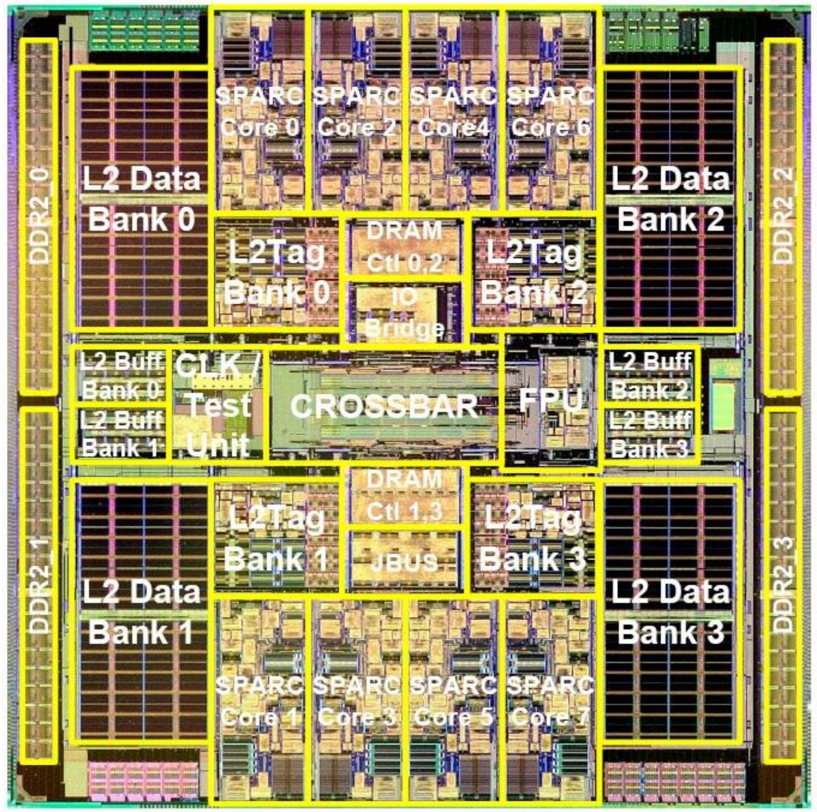
Uniform Access vs. Non-Uniform Access (NUMA)





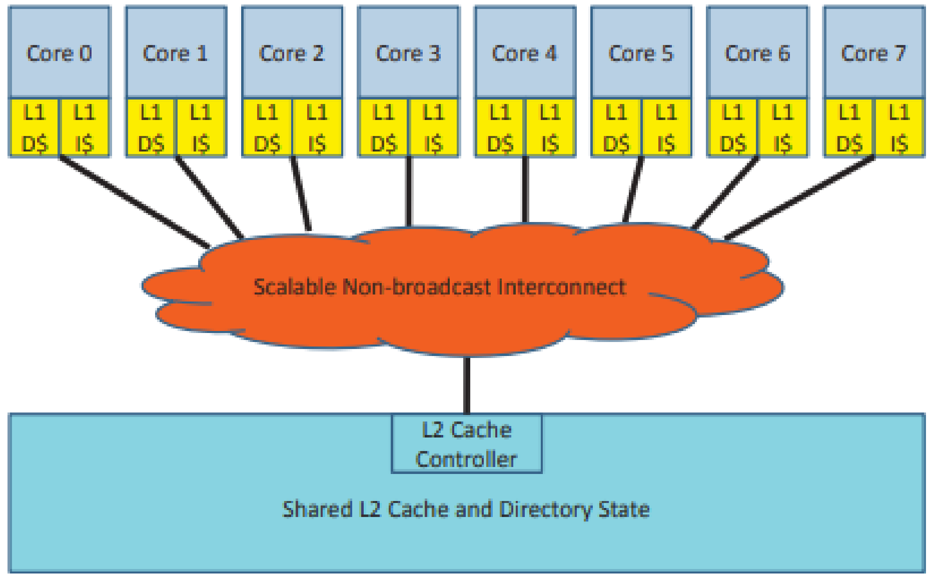
Centralized Last Level Cache

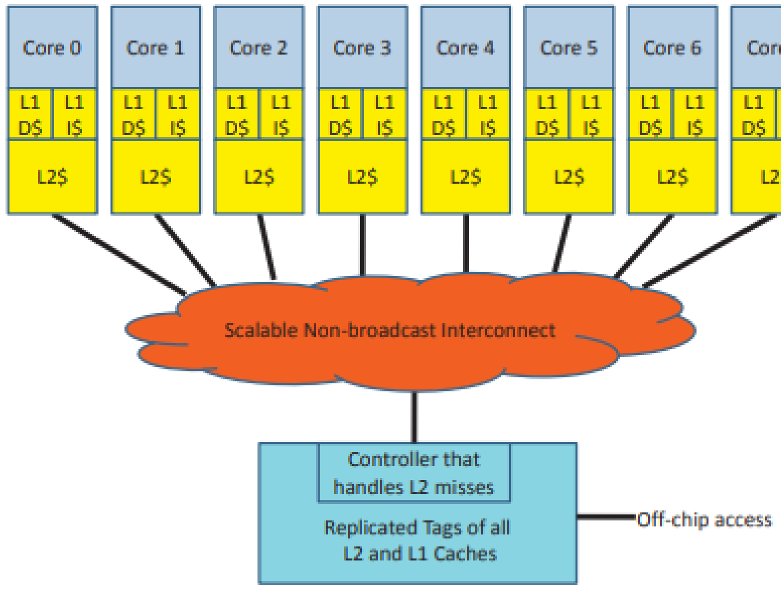




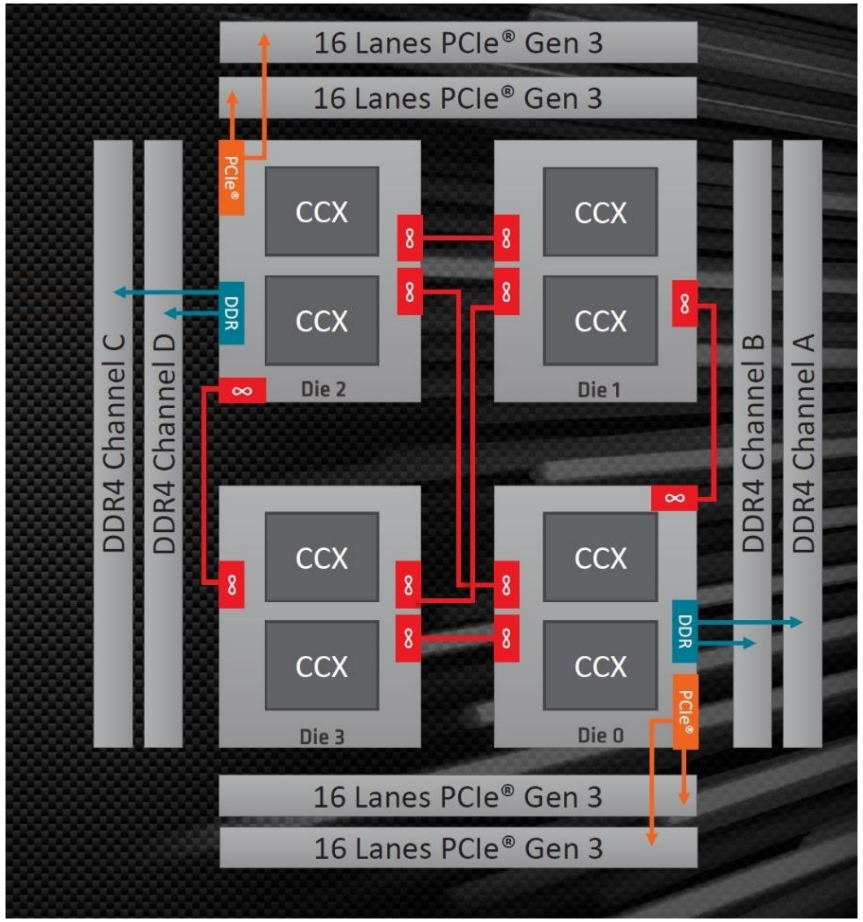
Xi-chuan Wang, Bin-fend Qian. The design of the cache crossbar based on OpenSPARC architecture 2008

Scalable Network for Shared Last Level Cache




Core 0
Core 1
Core 2
Core 3
Core 4
Core 5
Core 6
Core 7
L1 D$
L1 I$
L2$
Scalable Non-broadcast Interconnect
Controller that handles L2 misses
Replicated Tags of all L2 and L1 Caches
Off-chip access

AMD Zen Architecture's



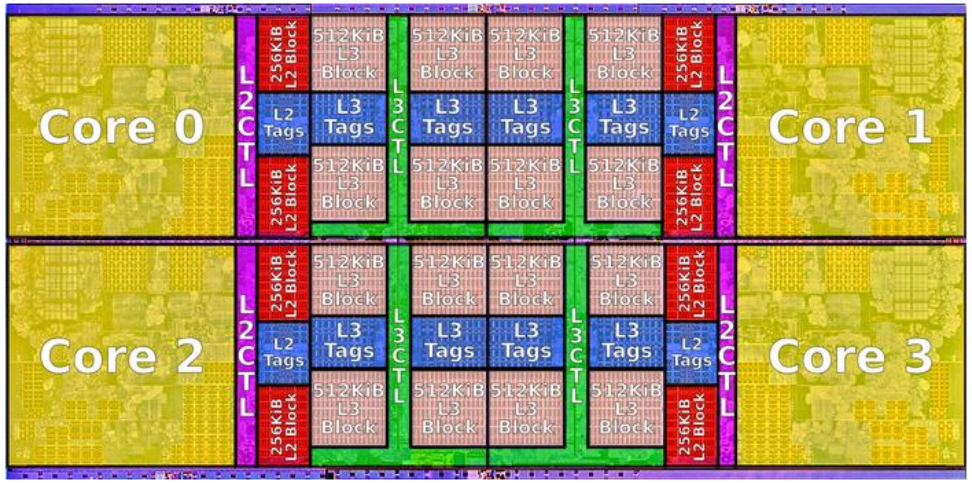


Figure from AMD

- ✓ Caches
- ➢ **Consistency**
- ➢ Coherence
- ➢ Victim Replication

Consistency Models

- Specification of the allowed behavior of multithreaded programs executing with shared memory
- Defines what orderings of distributed stores and loads are valid
- A memory system is consistent if any program on it gives allowed behavior.
- Often implemented through cache coherence
- Programming language can provide different model then hardware

- ✓ Caches
- ✓ Consistency
- ➢ **Coherence**
- ➢ Victim Replication

MESI coherence

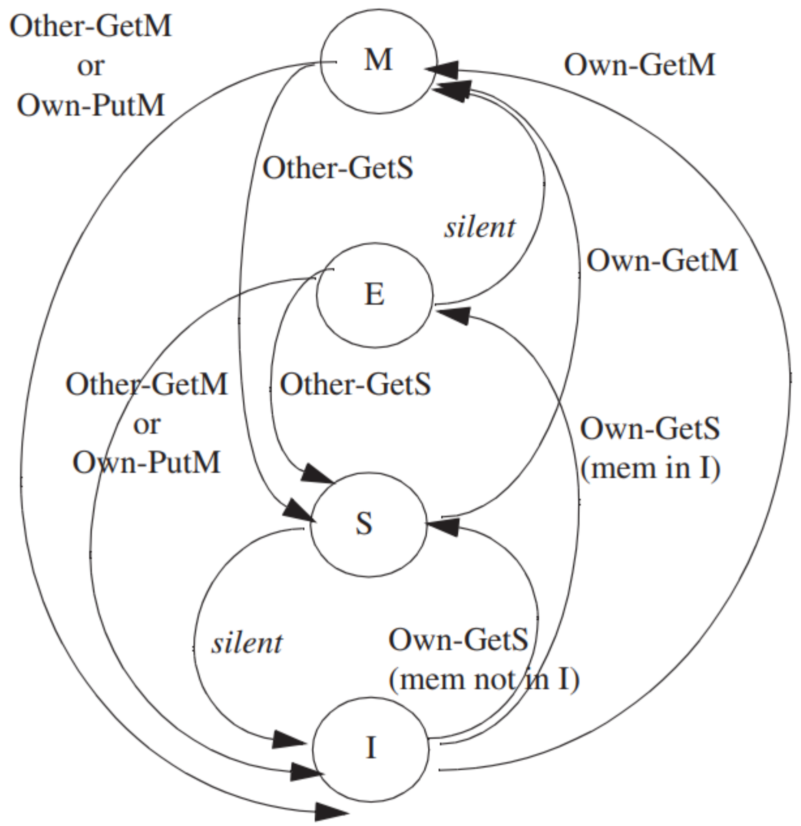


FIGURE 7.4: MESI: Transitions between stable states at cache controller

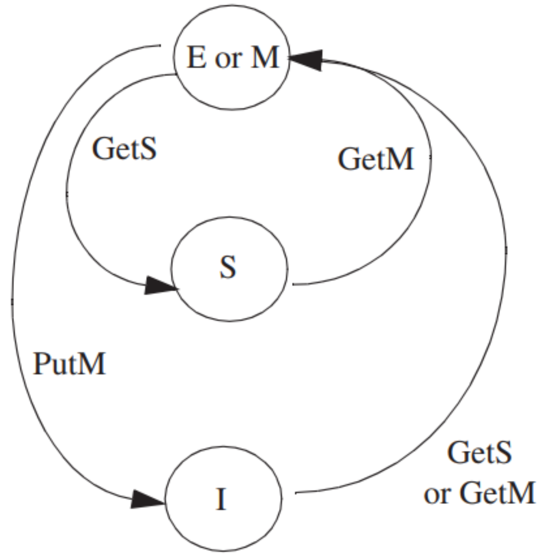
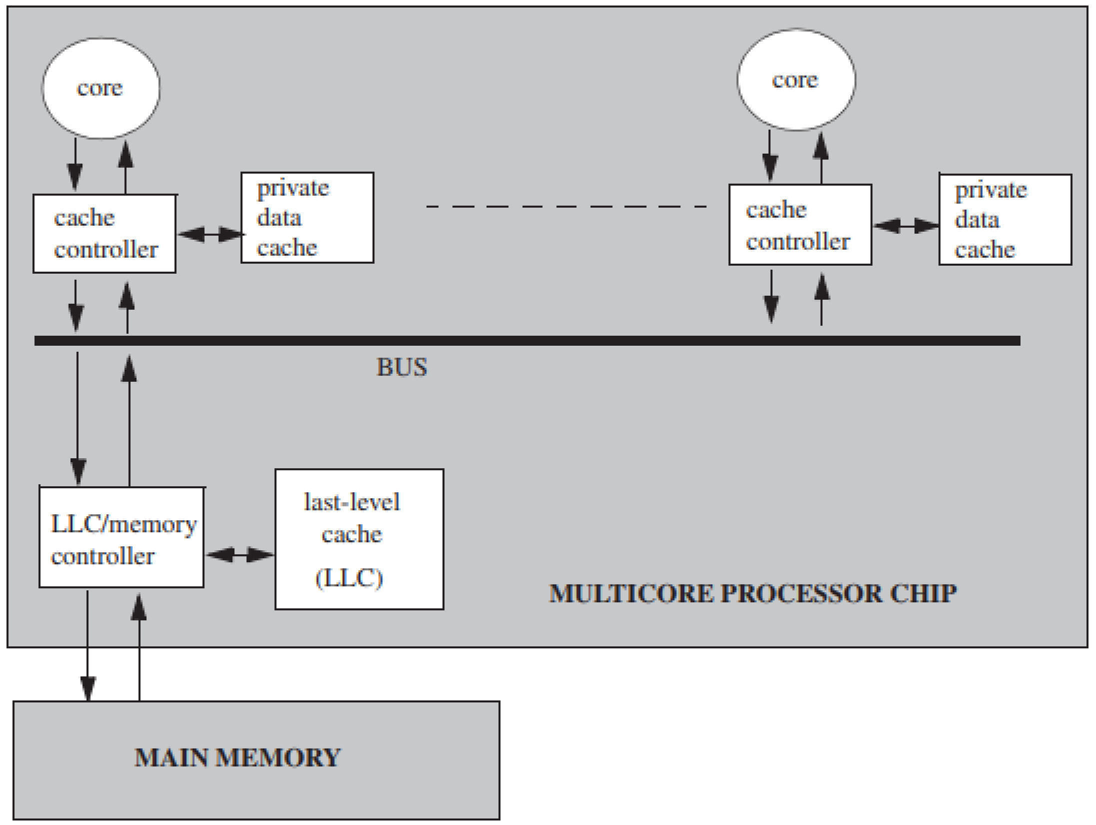


FIGURE 7.5: MESI: Transitions between stable states at memory controller

Other coherence protocols

- Lots of them!
  - MSI, MESI, MOSI, MOESI, MERSI, MESIF, write-once, Synapse, Berkely, Firefly, Dragon
- Software Coherence
  - FENCE operation
  - Evict operation

Snooping Coherence

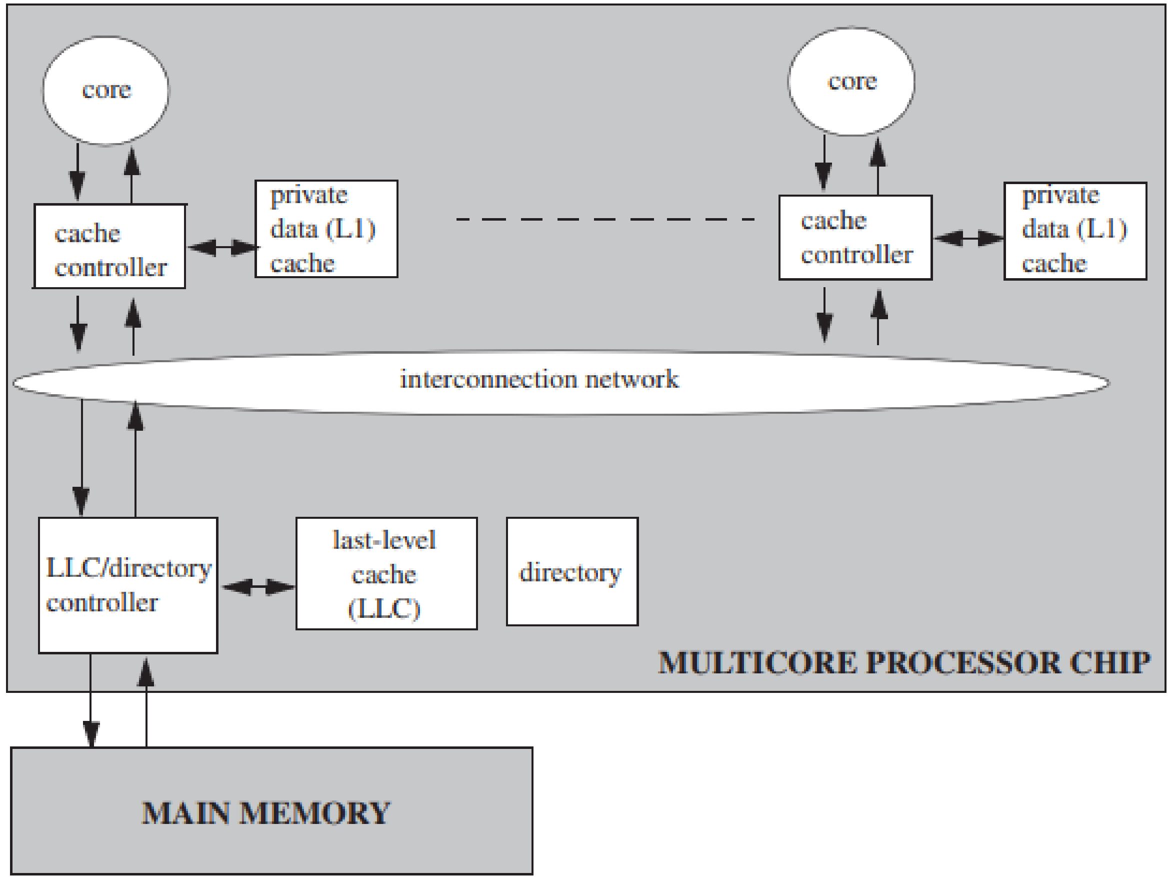


Power5 Snooping

- MESI snooping on split-transaction bus
- Nodes connected in unidirectional rings
- Message Types: Requests, Snoop response/Decision messages, Data
- Every request goes around the ring

- No shared bus, only point to point communication
- Use ring ordering to ensure consistency ordering

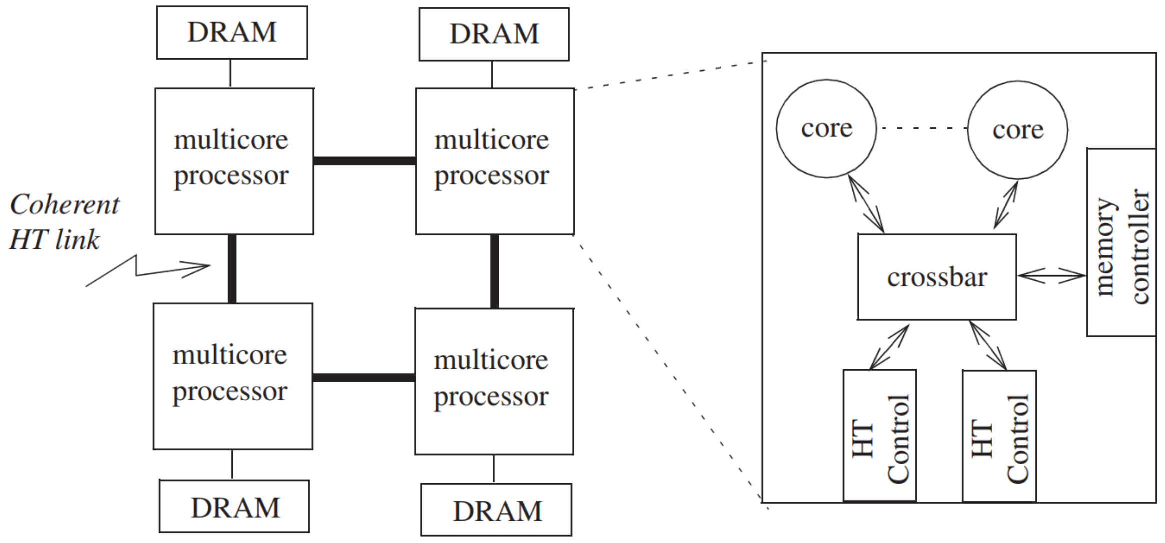
Directory Based Coherence



AMD’s Directory

- Zen
  - Distributed MOESI cache coherence Directory
  - Separate core complexes commination over “infinity fabric” network
  - No published information available

- Previous Generation AMD
  - Similar idea- and published!
  - Core requests to Directory Controller
  - Directory request state from cores
  - Responds to directory controller at home node

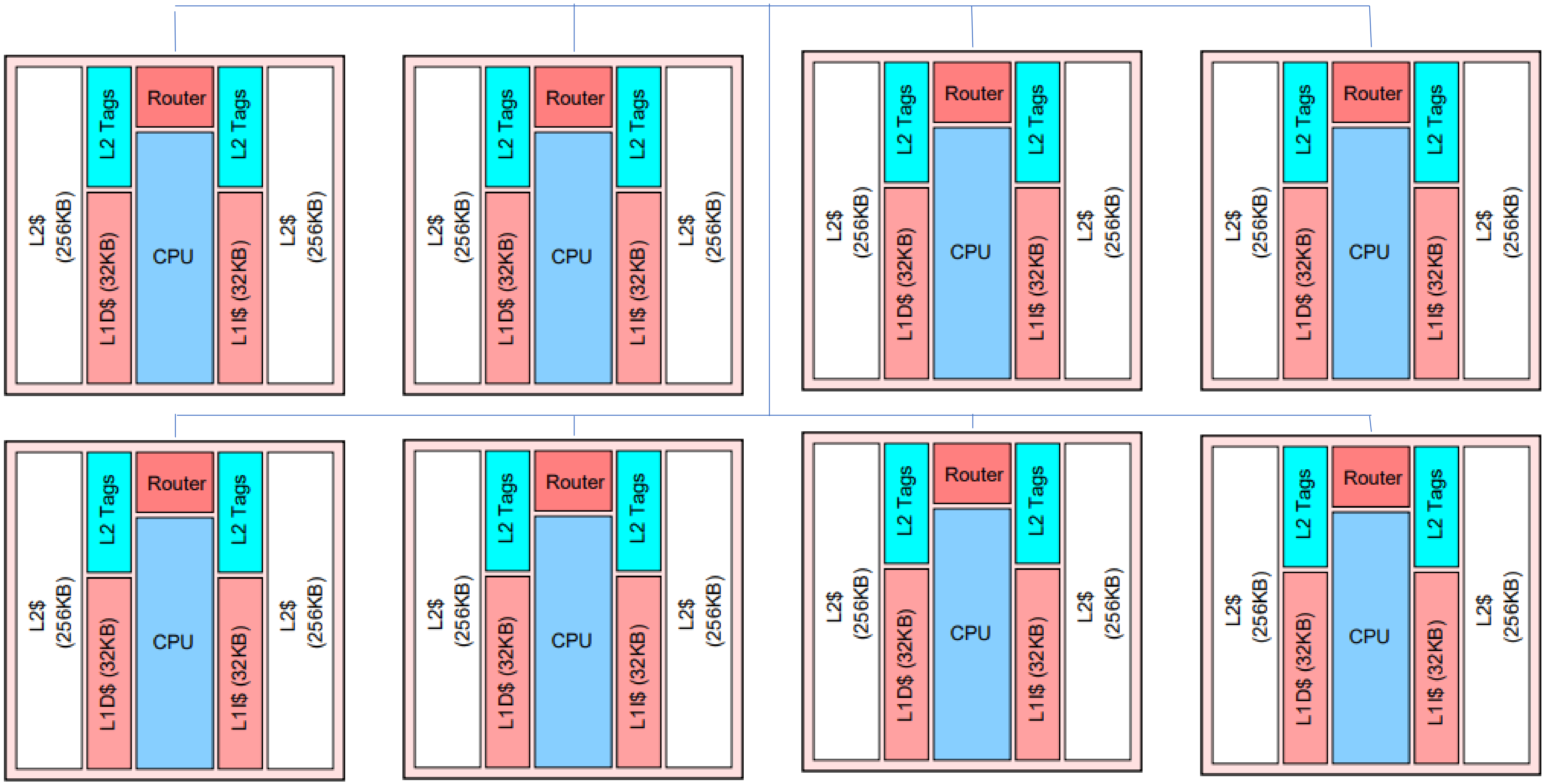


Complications in Practice

- What are some complications we might need to consider?
    - Out of Order Execution
    - Instruction Cache
    - Multi-level Cache
    - Write-through caches
    - Translation Lookaside Buffer (TLBs)
    - Direct Memory Access (DMA)
    - Virtual Caches
    - Hierarchical Coherence
    - Performance Issues

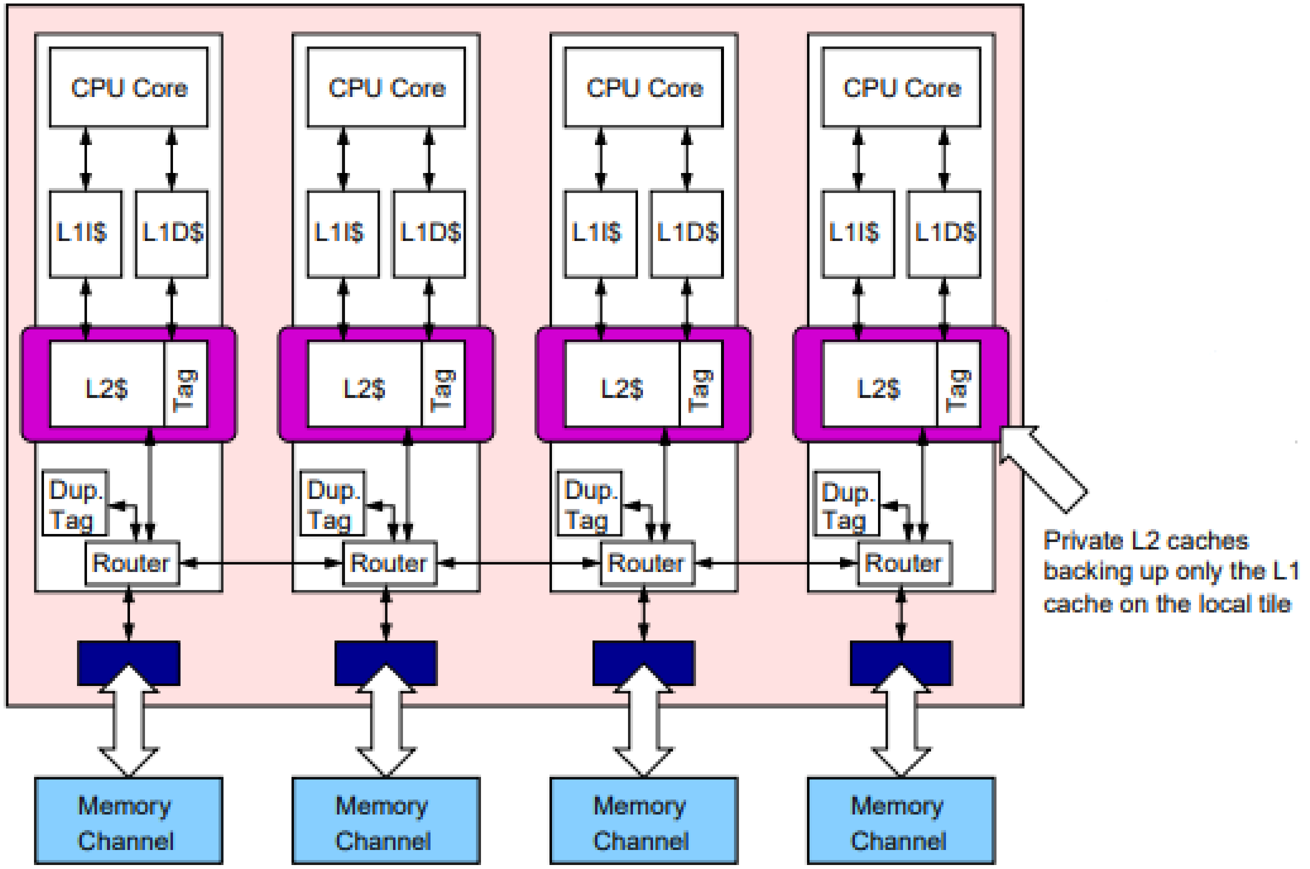
- ✓ Caches
- ✓ Consistency
- ✓ Coherence
- ➢ Victim Replication

Memory model



Michael Zhang and Krste Asanovic. Victim Replication: maximizing capacity while hiding wire delay in tiled chip multiprocessors 2005

Private L2 Cache

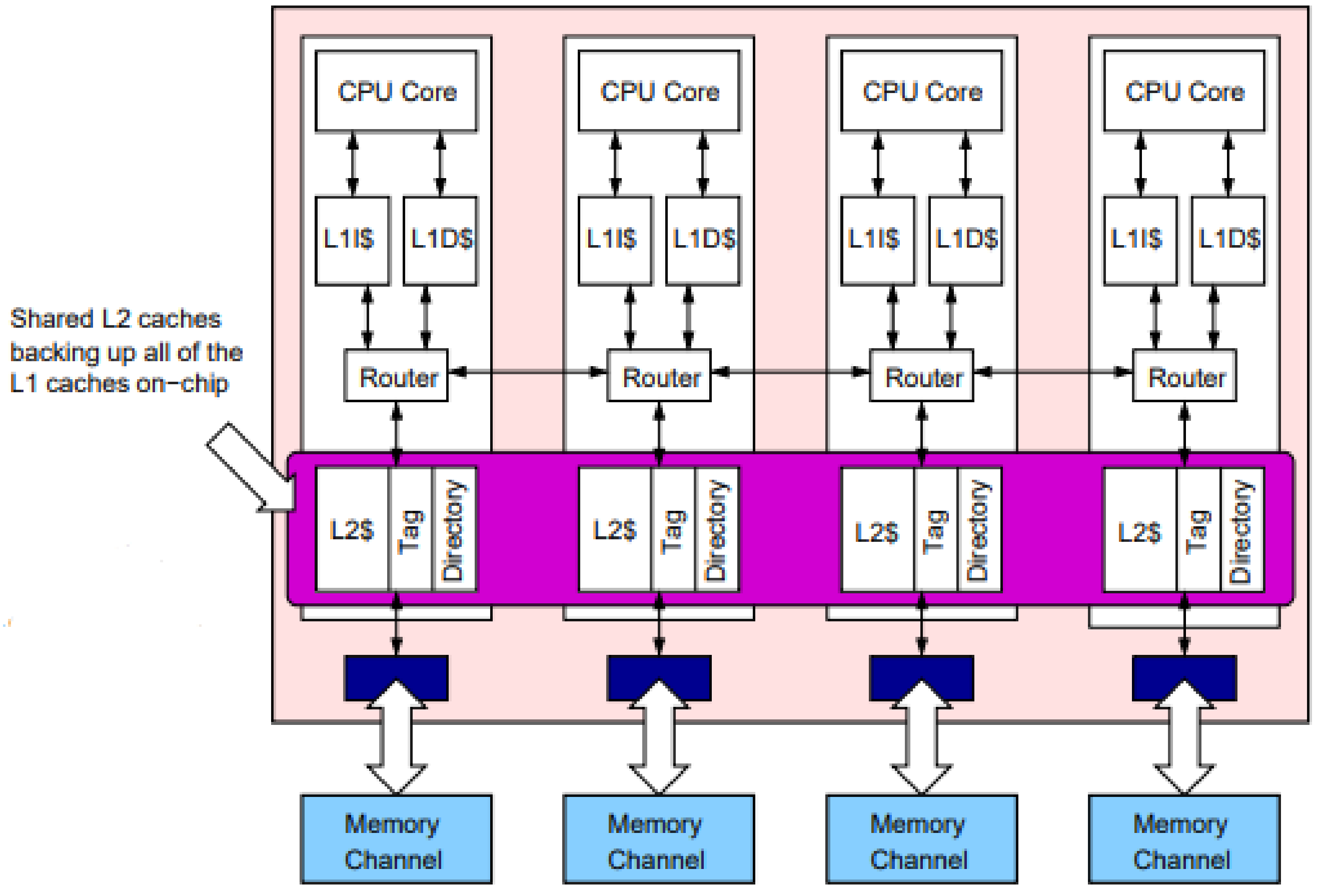


CPU Core
L1I$
L1D$
L2$
Tag
Dup. Tag
Router
Memory Channel
Private L2 caches backing up only the L1 cache on the local tile

Movement Caches

- Not in victim replication paper, but part of motivation
- Move data close consumer
  - Benefit: low latency
  - Cost: locating data requires complex logic

Shared L2 Cache



Victim Replication

- Need large shared cache like in L2S
- Home slices allow for fast unicast directory lookups
- When evicting from L1, can move to L2 without generating coherence message since directory already has it at local core
- Can get benefits of both large shared cache and smaller private cache.

Replication Policy

- L2VR replication policy will replace cache lines in this order
  - An invalid Line
  - Global line with no sharers
  - Existing Replica
- If all L2 is global and shared, doesn’t cache victim
- Within category is random
- Never replicates a victim with local home

Required Hardware Support

- L2P
  - Need to store full tag bits
- L2S
  - Don’t need to store tag bits for home tile, since data is always at home tile
- L2VR
  - Can discern victim cache with share with home bits
  - Tag width same as L2P – must hold tags from any tile

Experiment Design

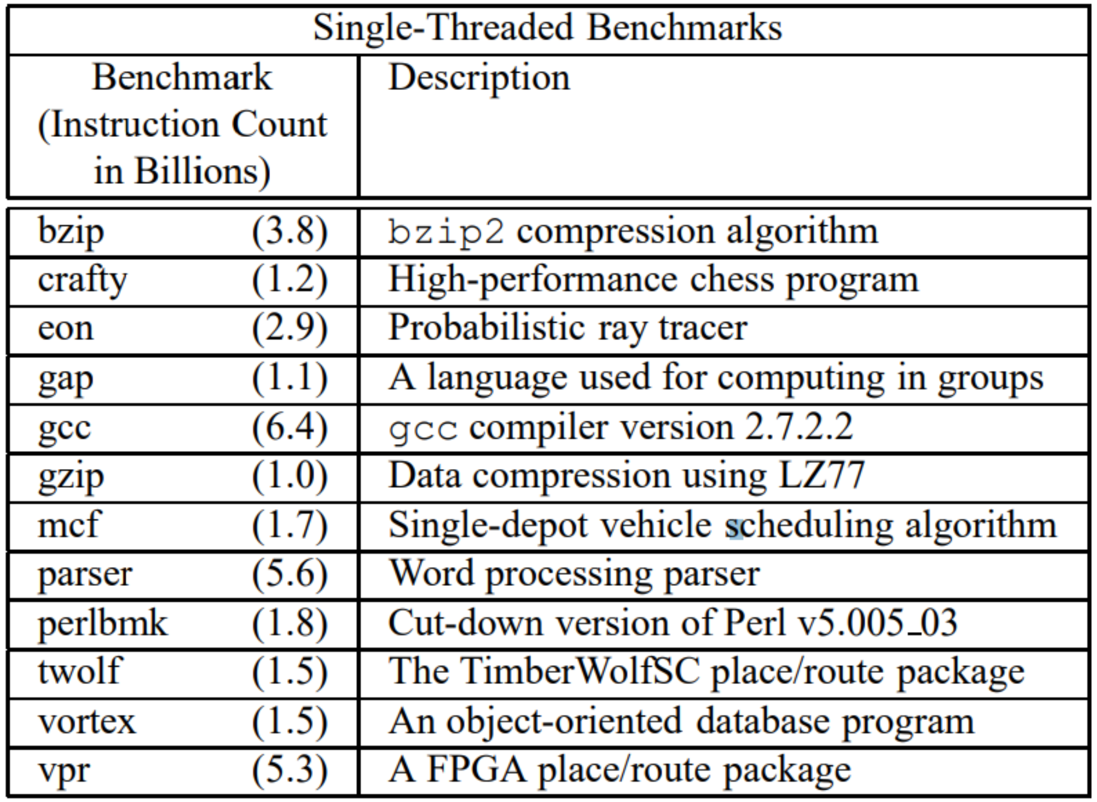
- Cache associativity
- Simulation sampling
- Benchmark suite

Limitations

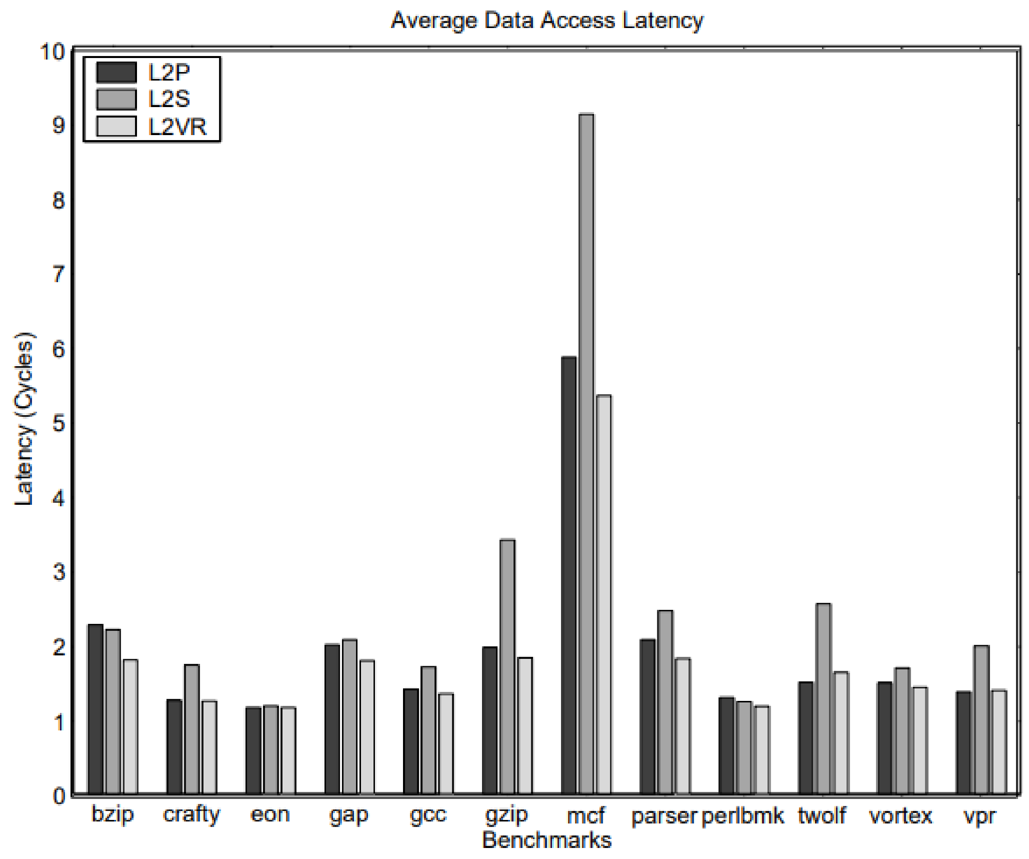
- What are some limitations of the paper's simulation?
  - Simple in-order processor, measure average memory latency
  - No consideration of prefetching, decoupling, non-blocking caches and out-of-order execution
  - Simulation only uses 8 processors
  - Set associativity

- Average memory latency should still mirror actual performance of a system with these techniques
- Reducing on chip traffic useful goal in its own right

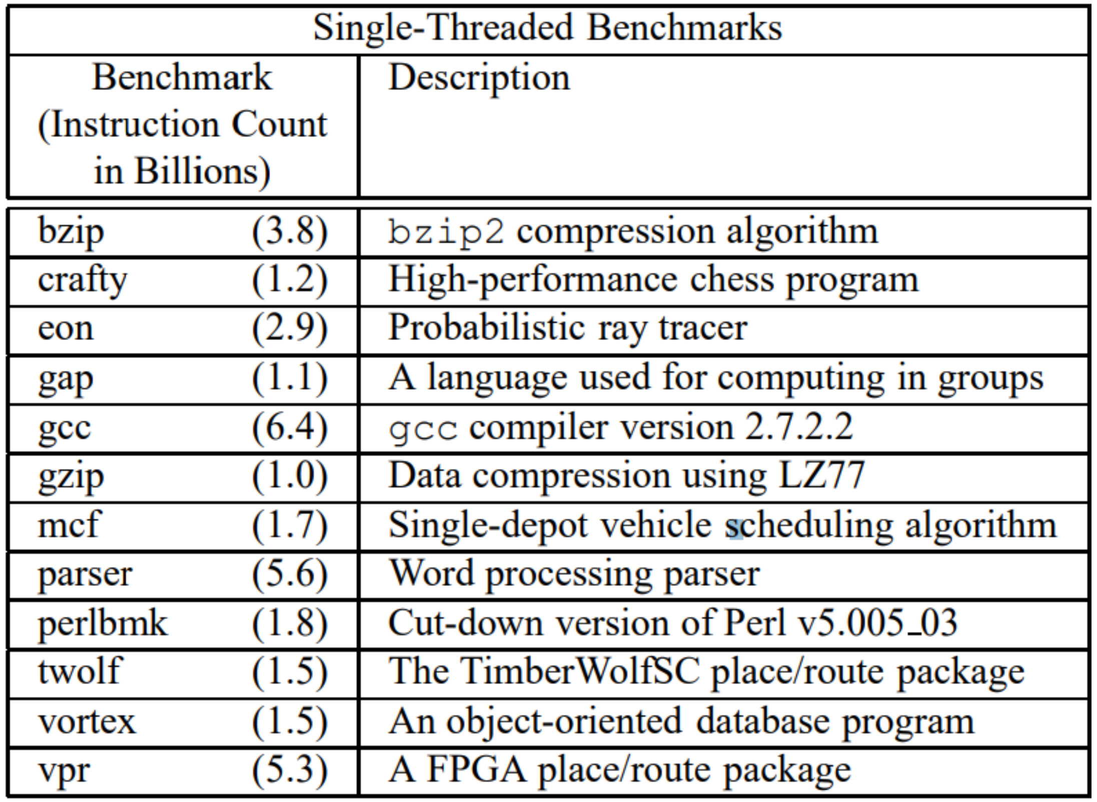
Single Threaded Benchmarks



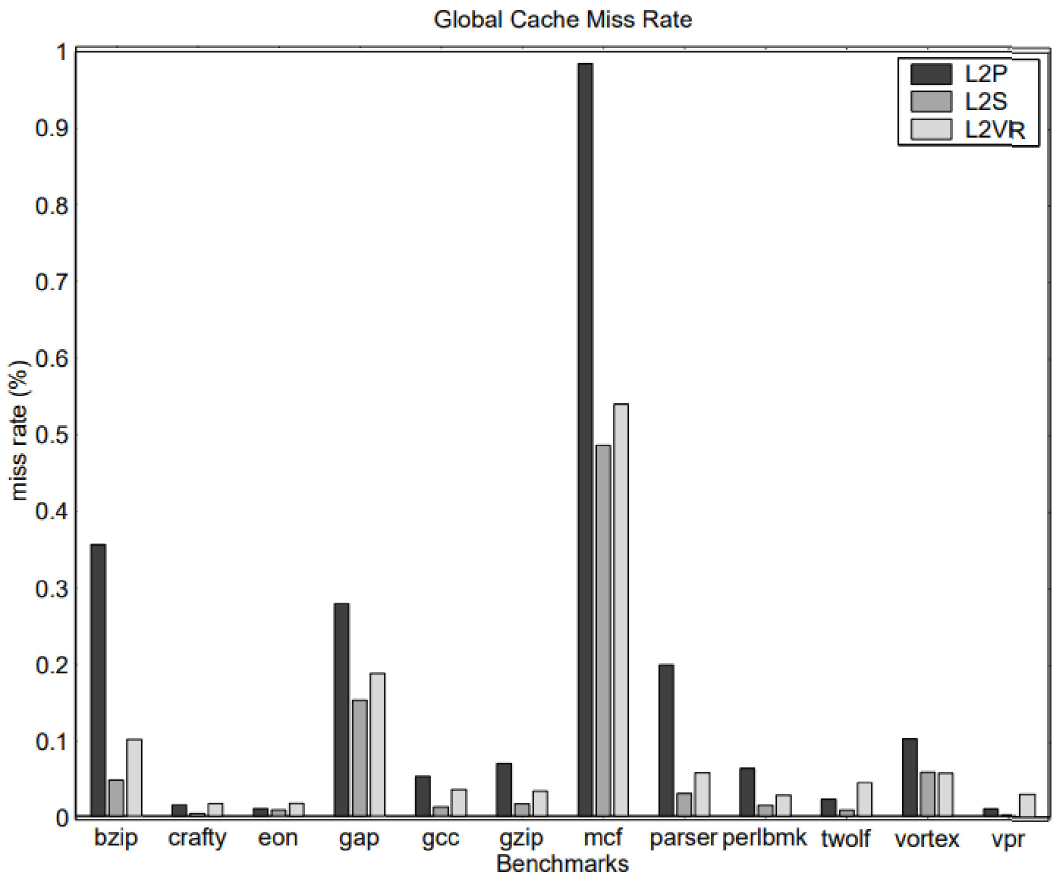
Single-Threaded Benchmarks	
Benchmark (Instruction Count in Billions)	Description
bzip (3.8)	`bzip2` compression algorithm
crafty (1.2)	High-performance chess program
eon (2.9)	Probabilistic ray tracer
gap (1.1)	A language used for computing in groups
gcc (6.4)	`gcc` compiler version 2.7.2.2
gzip (1.0)	Data compression using LZ77
mcf (1.7)	Single-depot vehicle scheduling algorithm
parser (5.6)	Word processing parser
perlbmk (1.8)	Cut-down version of Perl v5.005_03
twolf (1.5)	The TimberWolfSC place/route package
vortex (1.5)	An object-oriented database program
vpr (5.3)	A FPGA place/route package



Off Chip Miss Rate (Single Threaded)

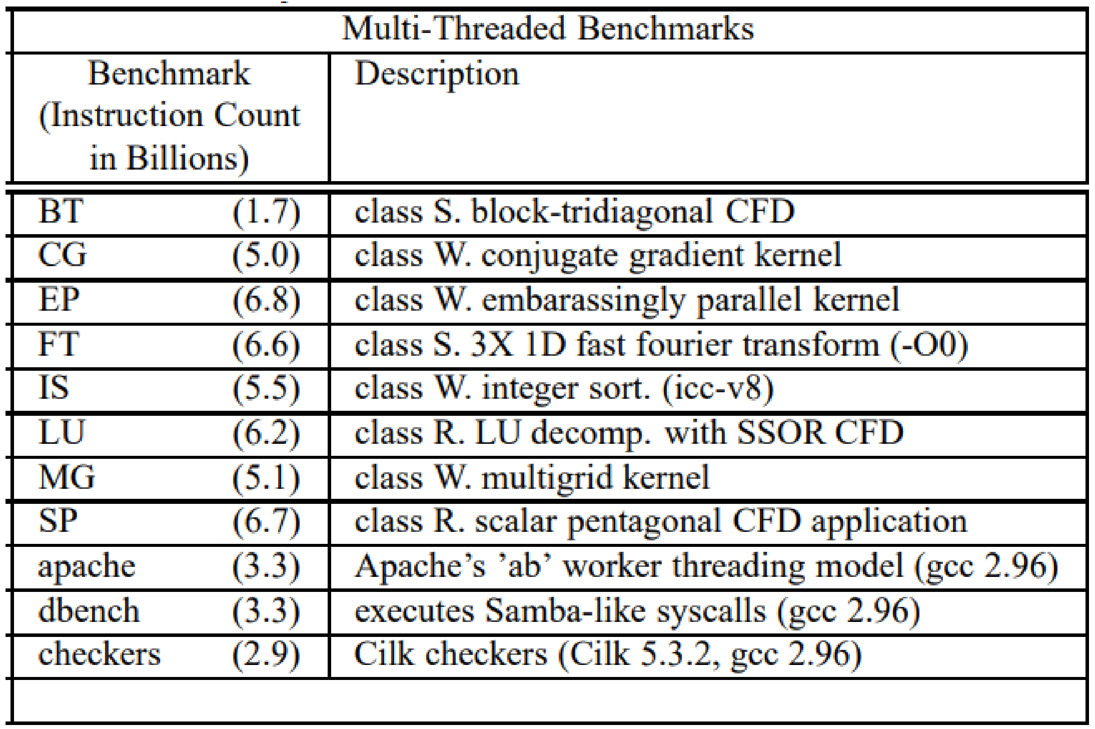


Single-Threaded Benchmarks		
Benchmark (Instruction Count in Billions)		Description
bzip	(3.8)	`bzip2` compression algorithm
crafty	(1.2)	High-performance chess program
eon	(2.9)	Probabilistic ray tracer
gap	(1.1)	A language used for computing in groups
gcc	(6.4)	`gcc` compiler version 2.7.2.2
gzip	(1.0)	Data compression using LZ77
mcf	(1.7)	Single-depot vehicle scheduling algorithm
parser	(5.6)	Word processing parser
perlbmk	(1.8)	Cut-down version of Perl v5.005_03
twolf	(1.5)	The TimberWolfSC place/route package
vortex	(1.5)	An object-oriented database program
vpr	(5.3)	A FPGA place/route package

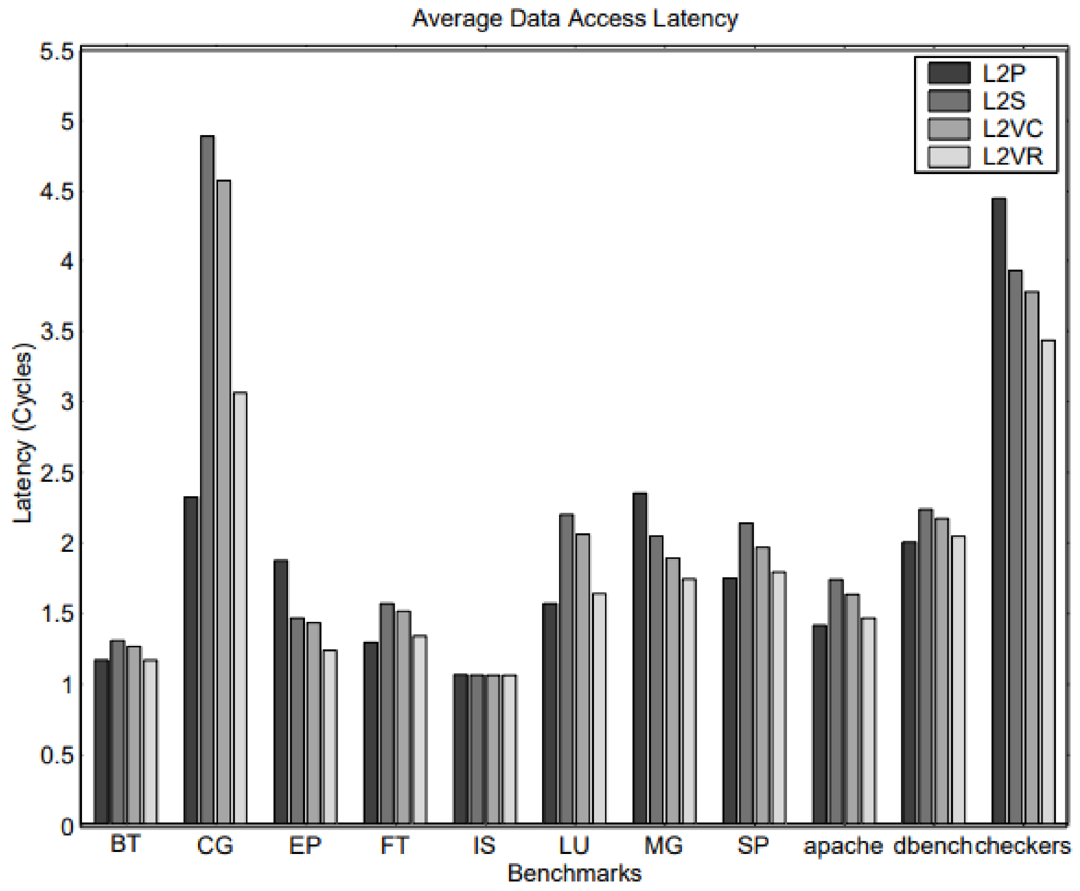


Michael Zhang and Krste Asanovic. Victim Replication: maximizing capacity while hiding wire delay in tiled chip multiprocessors 2005

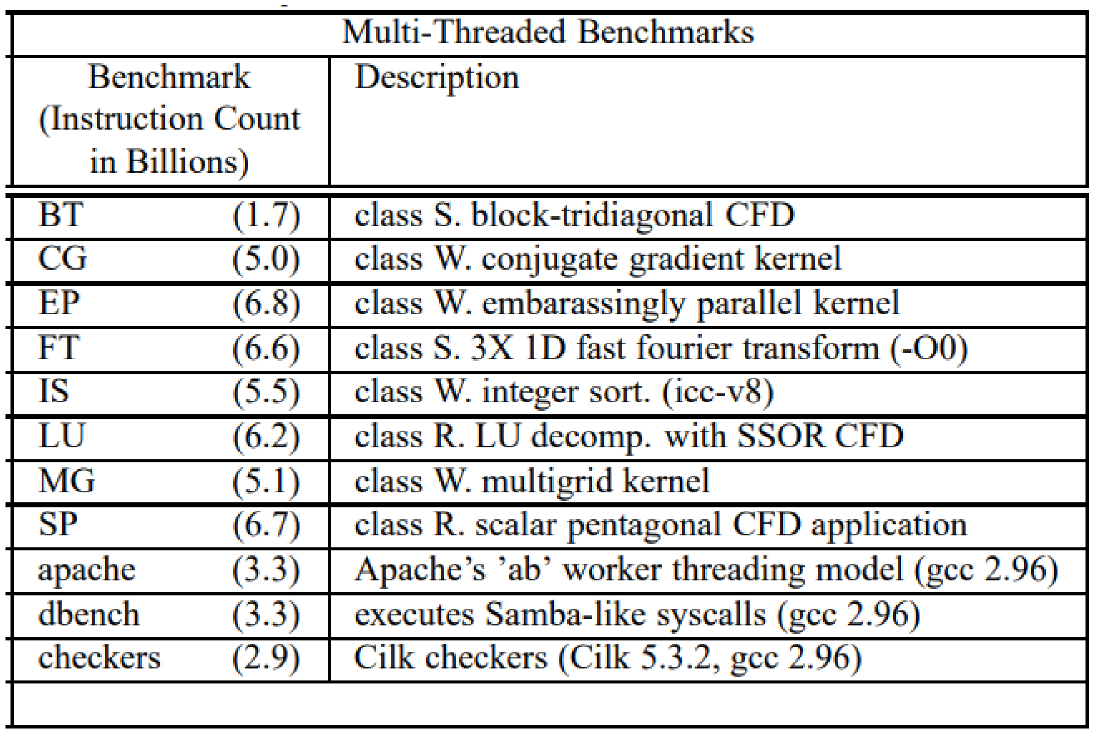
Multithreaded benchmarks



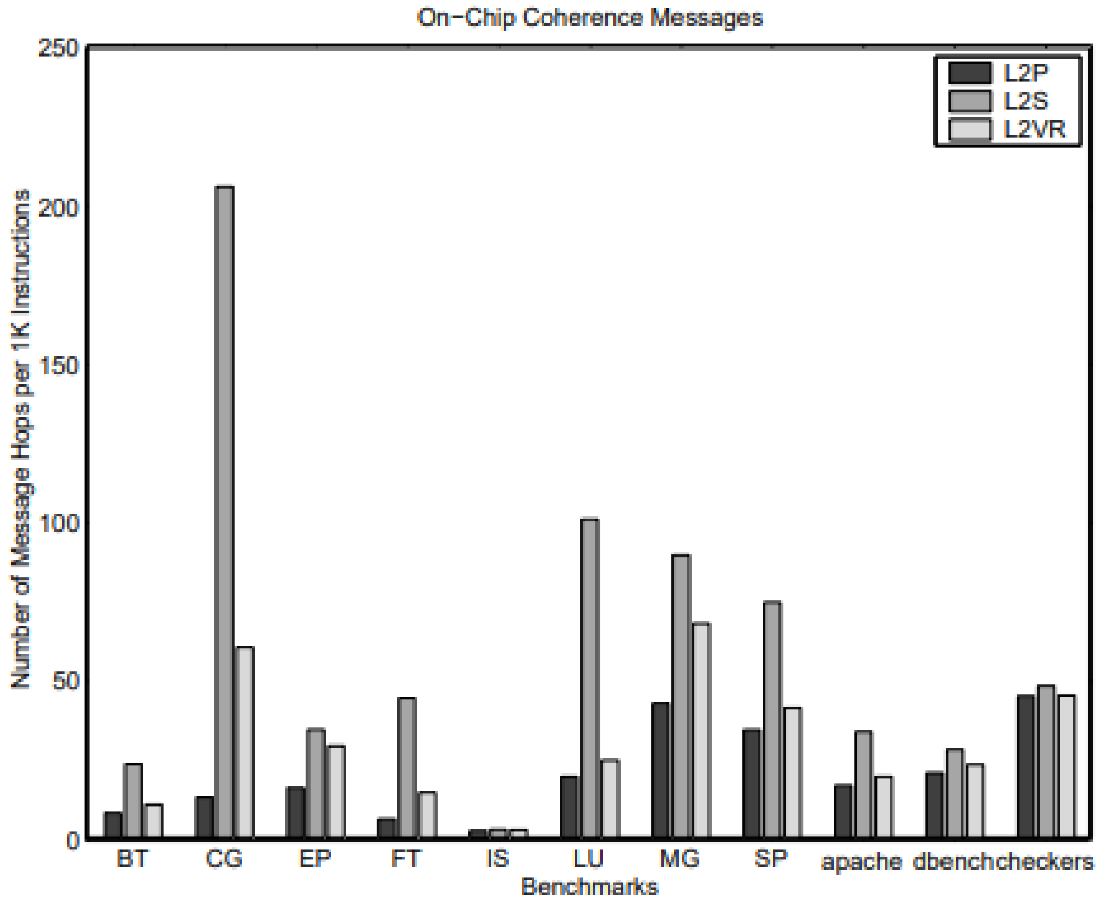
Multi-Threaded Benchmarks		
Benchmark (Instruction Count in Billions)		Description
BT	(1.7)	class S. block-tridiagonal CFD
CG	(5.0)	class W. conjugate gradient kernel
EP	(6.8)	class W. embarassingly parallel kernel
FT	(6.6)	class S. 3X 1D fast fourier transform (-O0)
IS	(5.5)	class W. integer sort. (icc-v8)
LU	(6.2)	class R. LU decomp. with SSOR CFD
MG	(5.1)	class W. multigrid kernel
SP	(6.7)	class R. scalar pentagonal CFD application
apache	(3.3)	Apache's 'ab' worker threading model (gcc 2.96)
dbench	(3.3)	executes Samba-like syscalls (gcc 2.96)
checkers	(2.9)	Cilk checkers (Cilk 5.3.2, gcc 2.96)



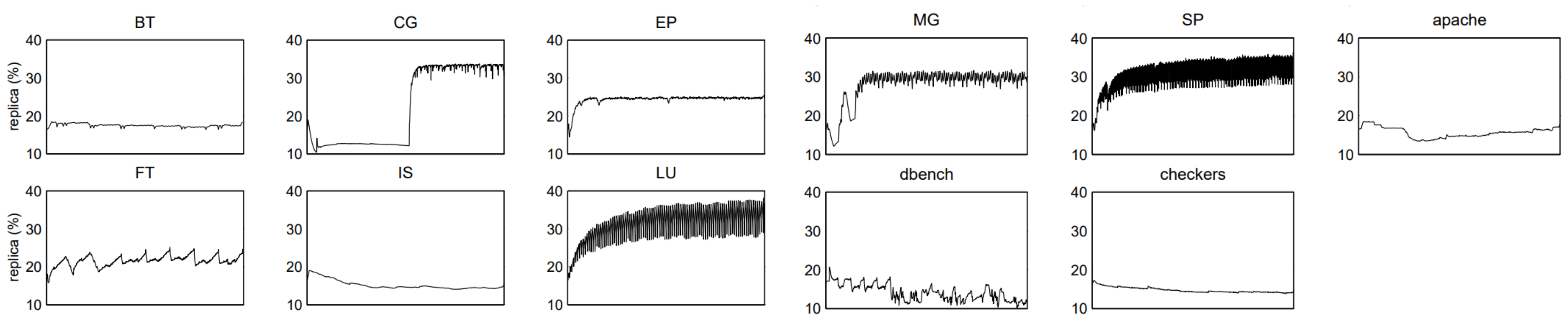
On Chip Network Messages (multithreaded)



Multi-Threaded Benchmarks		
Benchmark (Instruction Count in Billions)		Description
BT	(1.7)	class S. block-tridiagonal CFD
CG	(5.0)	class W. conjugate gradient kernel
EP	(6.8)	class W. embarassingly parallel kernel
FT	(6.6)	class S. 3X 1D fast fourier transform (-O0)
IS	(5.5)	class W. integer sort. (icc-v8)
LU	(6.2)	class R. LU decomp. with SSOR CFD
MG	(5.1)	class W. multigrid kernel
SP	(6.7)	class R. scalar pentagonal CFD application
apache	(3.3)	Apache's 'ab' worker threading model (gcc 2.96)
dbench	(3.3)	executes Samba-like syscalls (gcc 2.96)
checkers	(2.9)	Cilk checkers (Cilk 5.3.2, gcc 2.96)



L2VR allocation



Victim Replication Performance

- When is L2S better?
- When is L2P better?
- What costs does L2VR have?
- Is the victim replacement policy optimal?

IBM Power7 Architecture

- Adaptive Victim L3 Cache
  - Each core has 4 MG local region
  - Adaptive cache policy routes data to L3 region close to cores that use them
  - Directory has 13 states, L3 cache policy works with these states to minimize coherence messages
- On L2 miss, goes to local L3 region
  - On local L3 miss, is broadcasts on coherence fabric, snooped by other L2/L3s
- Datum evicted from L2 go into L3 under similar circumstances as the paper
- L3 associativity improved by utilizing multiple L3 caches, rather then predefined “home” slices as in paper

AMD Zen Architecture

- L3 Cache is A Victim Cache
    - CCX level granularity
- Similar on chip network to Power for directory based coherence

- ✓ Caches
- ✓ Consistency
- ✓ Coherence
- ✓ Victim Replication

Some other papers to check out

- https://doi.org/10.1109/ISCA.2005.39
- https://doi.org/10.1109/MICRO.2006.10
- https://doi.org/10.1145/1150019.1136509