Advanced Caches

Prof. Mikko H. Lipasti
University of Wisconsin-Madison

Lecture notes based on notes by John P. Shen and Mark Hill
Updated by Mikko Lipasti

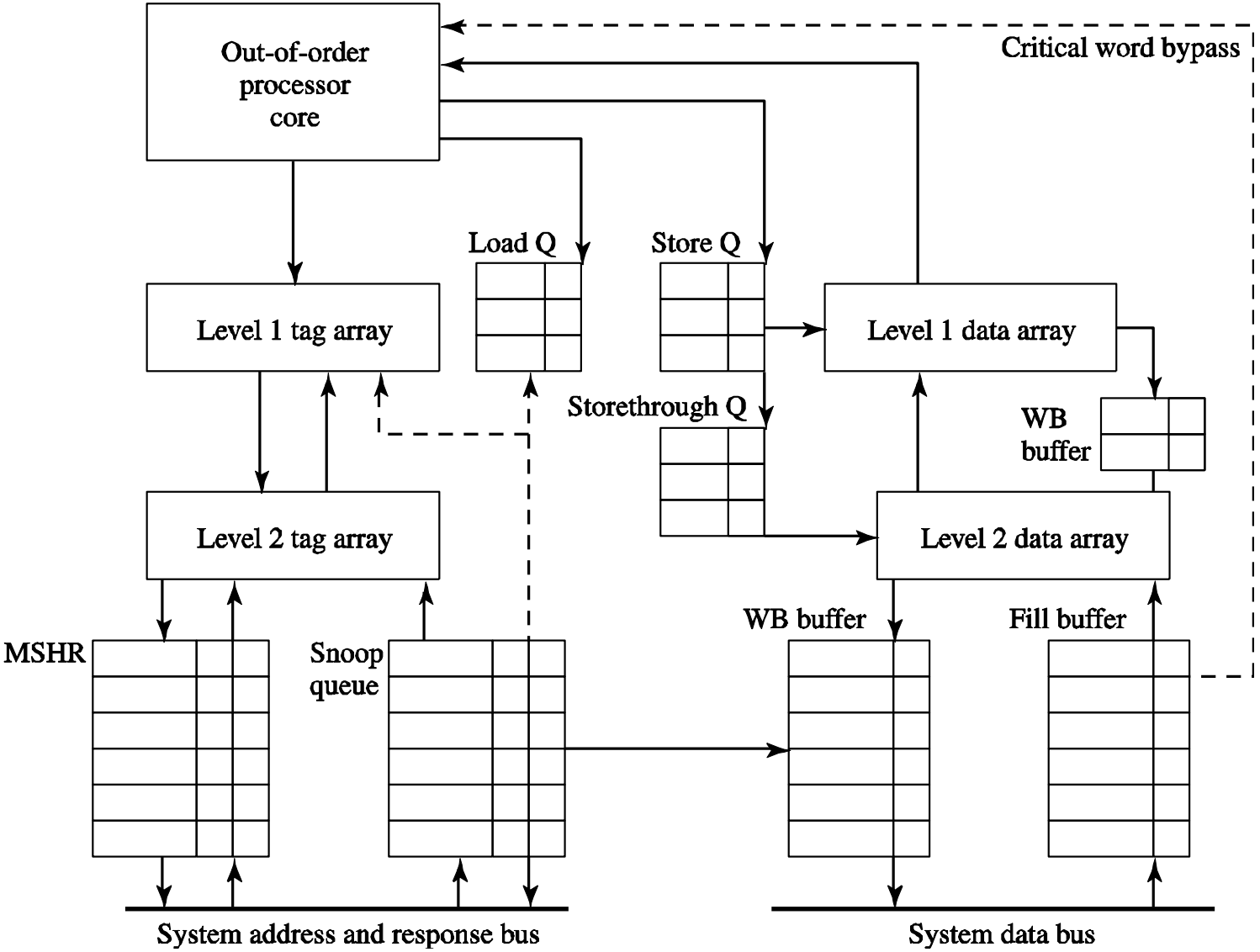
Readings

- Read on your own:
  - Review: Shen & Lipasti Chapter 3
  - W.-H. Wang, J.-L. Baer, and H. M. Levy. Organization of a two-level virtual-real cache hierarchy, *Proc. 16th ISCA,* pp. 140-148, June 1989 **(B6)** Online PDF
  - D. Kroft. Lockup-Free Instruction Fetch/Prefetch Cache Organization, *Proc. International Symposium on Computer Architecture* , May 1981 **(B6)**. Online PDF
  - Bruce Jacob, "The Memory System: You Can't Avoid It, You Can't Ignore It, You Can't Fake It," Synthesis Lectures on Computer Architecture 2009 4:1, 1-77. Online PDF
- Review:
  - Andreas Sembrant, Erik Hagersten, David Black-Schaffer, "The Direct-to-Data (D2D) cache: navigating the cache hierarchy with a single lookup," Proc. ISCA 2014, June 2014. Online PDF
  - Steven Pelley, Peter Chen, and Thomas Wenisch, "Memory Persistency," Proc. ISCA 2014, June 2014. Online PDF

Advanced Memory Hierarchy

- Coherent Memory Interface
- Evaluation methods
- Better miss rate: skewed associative caches, victim caches
- Reducing miss costs through software restructuring
- Higher bandwidth: Lock-up free caches, superscalar caches
- Beyond simple blocks
- Two level caches

Coherent Memory Interface



Coherent Memory Interface

- Load Queue
  - Tracks inflight loads for aliasing, coherence
- Store Queue
  - Defers stores until commit, tracks aliasing
- Storethrough Queue or Write Buffer or Store Buffer
  - Defers stores, coalesces writes, must handle RAW
- MSHR
  - Tracks outstanding misses, enables *lockup-free caches* [Kroft ISCA 91]
- Snoop Queue
  - Buffers, tracks incoming requests from coherent I/O, other processors
- Fill Buffer
  - Works with MSHR to hold incoming partial lines
- Writeback Buffer
  - Defers writeback of evicted line (demand miss handled first)

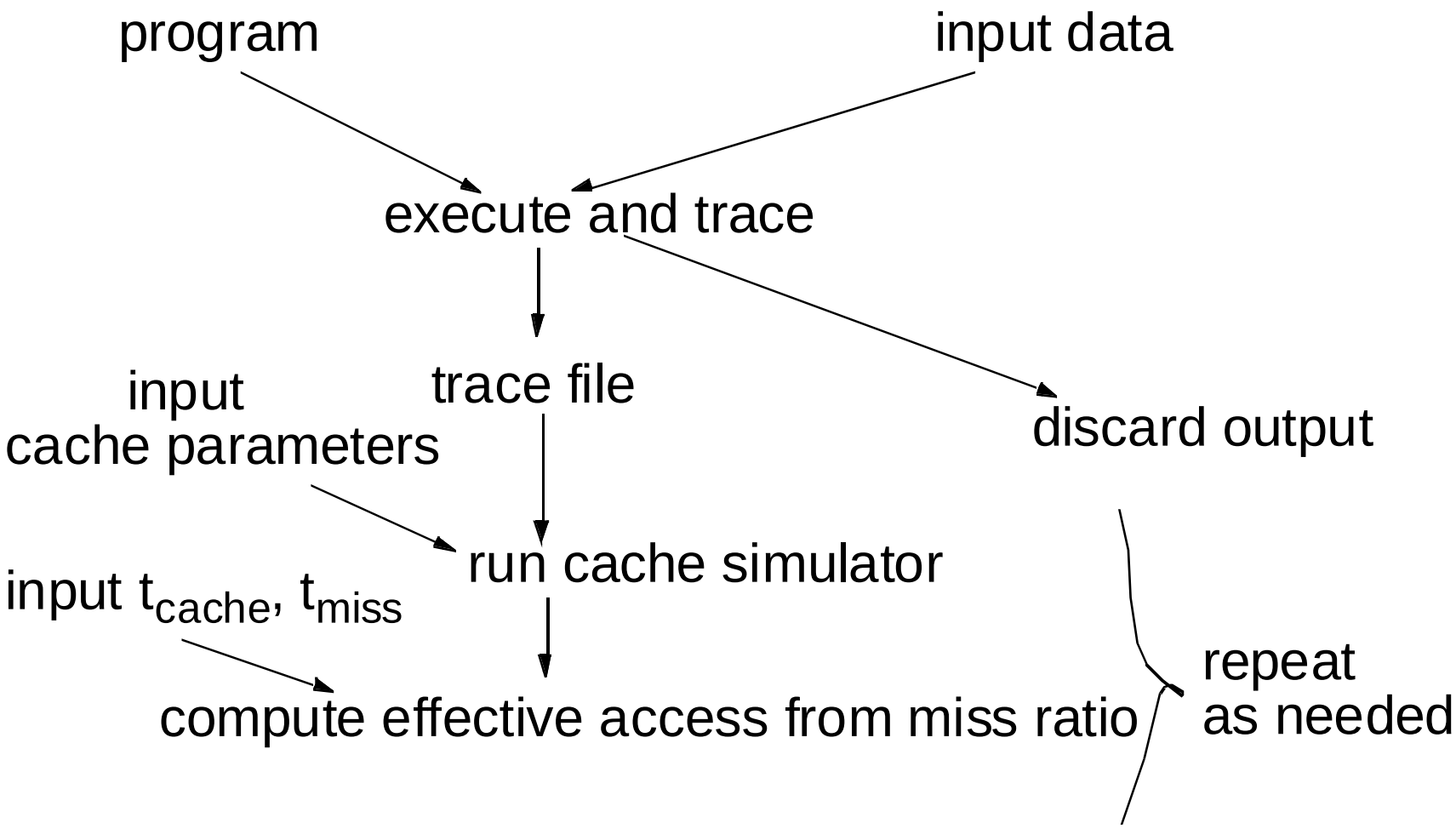
Evaluation Methods - Counters

- Counts hits and misses in hardware
  - see [Clark, TOCS 1983]
  - Intel VTune tool
- Accurate
- Realistic workloads - system, user, everything
- Requires machine to exist
- Hard to vary cache parameters
- Experiments not deterministic

Evaluation Methods - Analytical

- Mathematical expressions
  - Insight - can vary parameters
  - Fast
  - Absolute accuracy suspect for models with few parameters
  - Hard to determine many parameter values
  - Not widely used today

Evaluation: Trace-Driven Simulation



Evaluation: Trace-Driven Simulation

- Experiments repeatable
- Can be accurate
- Much recent progress
- Reasonable traces are very large ~gigabytes
- Simulation can be time consuming
- Hard to say if traces representative
- Don’t model speculative execution

Evaluation: Execution-Driven Simulation

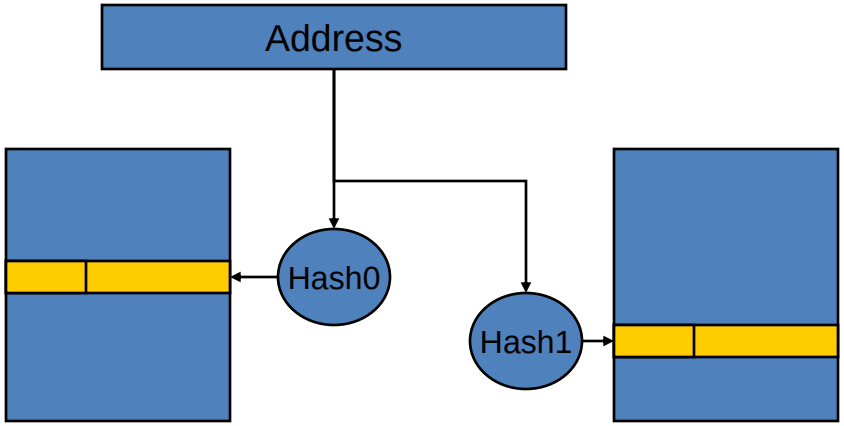
- Do full processor simulation each time
  - Actual performance; with ILP miss rate means nothing
    - Non-blocking caches
    - Prefetches (timeliness)
    - Pollution effects due to speculation
  - No need to store trace
  - Much more complicated simulation model
- Time-consuming - but good programming can help
- Very common today

Advanced Memory Hierarchy

- Coherent Memory Interface
- Evaluation methods
- **Better miss rate: skewed associative caches, victim caches**
- Reducing miss costs through software restructuring
- Higher bandwidth: Lock-up free caches, superscalar caches
- Beyond simple blocks
- Two level caches

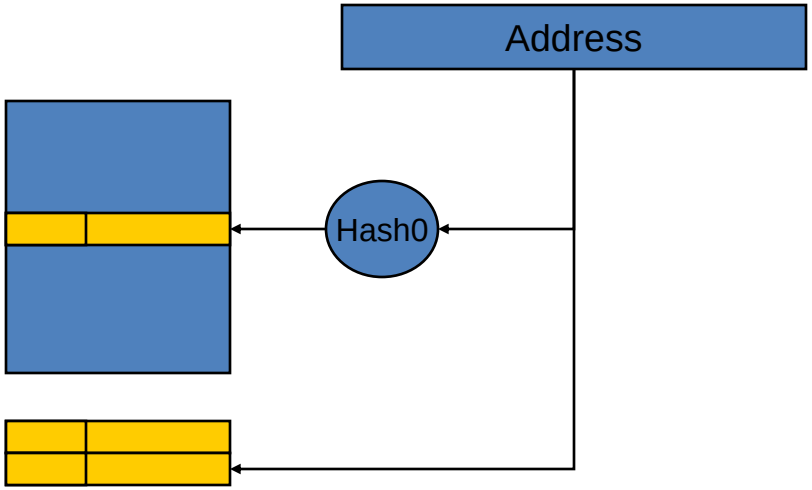
Seznec's Skewed Associative Cache

- Alleviates conflict misses in a conventional set assoc cache
- If two addresses conflict in 1 bank, they conflict in the others too
  - e.g., 3 addresses with same index bits will thrash in 2-way cache
- Solution: use different hash functions for each bank
- Works reasonably well: more robust conflict miss behavior
- But: how do you implement replacement policy?



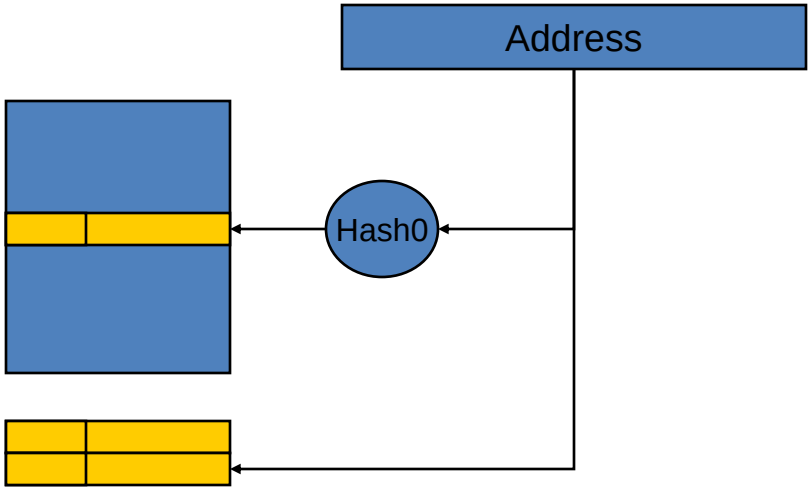
Jouppi's Victim Cache

- **Targeted at conflict misses**
- **Victim cache: a small fully associative cache**
  - **holds victims replaced in direct-mapped or low-assoc**
  - **LRU replacement**
  - **a miss in cache + a hit in victim cache**
    - **=> move line to main cache**
- **Poor man's associativity**
  - **Not all sets suffer conflicts; provide limited capacity for conflicts**



Jouppi's Victim Cache

- **Removes conflict misses, mostly useful for DM or 2-way**
  - **Even one entry helps some benchmarks**
  - **I-cache helped more than D-cache**
- **Versus cache size**
  - **Generally, victim cache helps more for smaller caches**
- **Versus line size**
  - **helps more with larger line size (why?)**
- **Used in Pentium Pro (P6) I-cache**



Advanced Memory Hierarchy

- Coherent Memory Interface
- Evaluation methods
- Better miss rate: skewed associative caches, victim caches
- **Reducing miss costs through software restructuring**
- Higher bandwidth: Lock-up free caches, superscalar caches
- Beyond simple blocks
- Two level caches
- Prefetching, software prefetching
- Main Memory, DRAM
- Virtual Memory, TLBs
- Interaction of caches, virtual memory

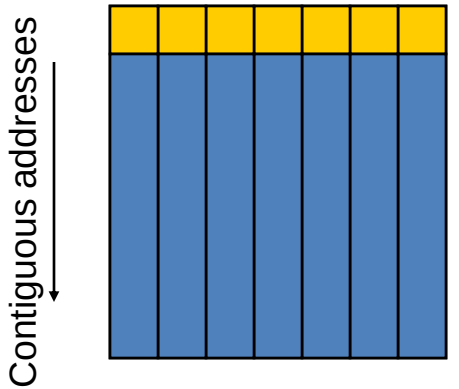
Software Restructuring

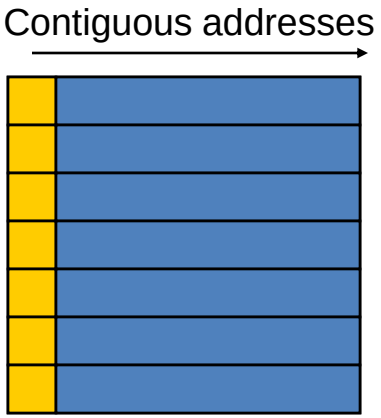
- **If column-major (Fortran)**
  - **x[i+1, j] follows x [i,j] in memory**
  - **x[i,j+1] long after x[i,j] in memory**
- **Poor code**

```
for i = 1, rows
  for j = 1, columns
    sum = sum + x[i,j]
```

- **Conversely, if row-major (C/C++)**
- **Poor code**

```
for j = 1, columns
  for i = 1, rows
    sum = sum + x[i,j]
```



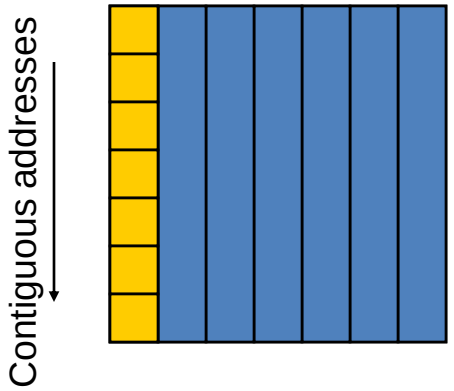


Software Restructuring

- **Better column-major code**

```
for j = 1, columns
  for i = 1, rows
    sum = sum + x[i,j]
```

- **Optimizations - need to check if it is valid to do them**
  - **Loop interchange (used above)**
  - **Blocking**
  - **Etc.**
- **Hard:** pointers, indirection, unknown loop bounds, sparse matrices



Advanced Memory Hierarchy

- Coherent Memory Interface
- Evaluation methods
- Better miss rate: skewed associative caches, victim caches
- Reducing miss costs through software restructuring
- **Higher bandwidth: Lock-up free caches, superscalar caches**
- Beyond simple blocks
- Two level caches

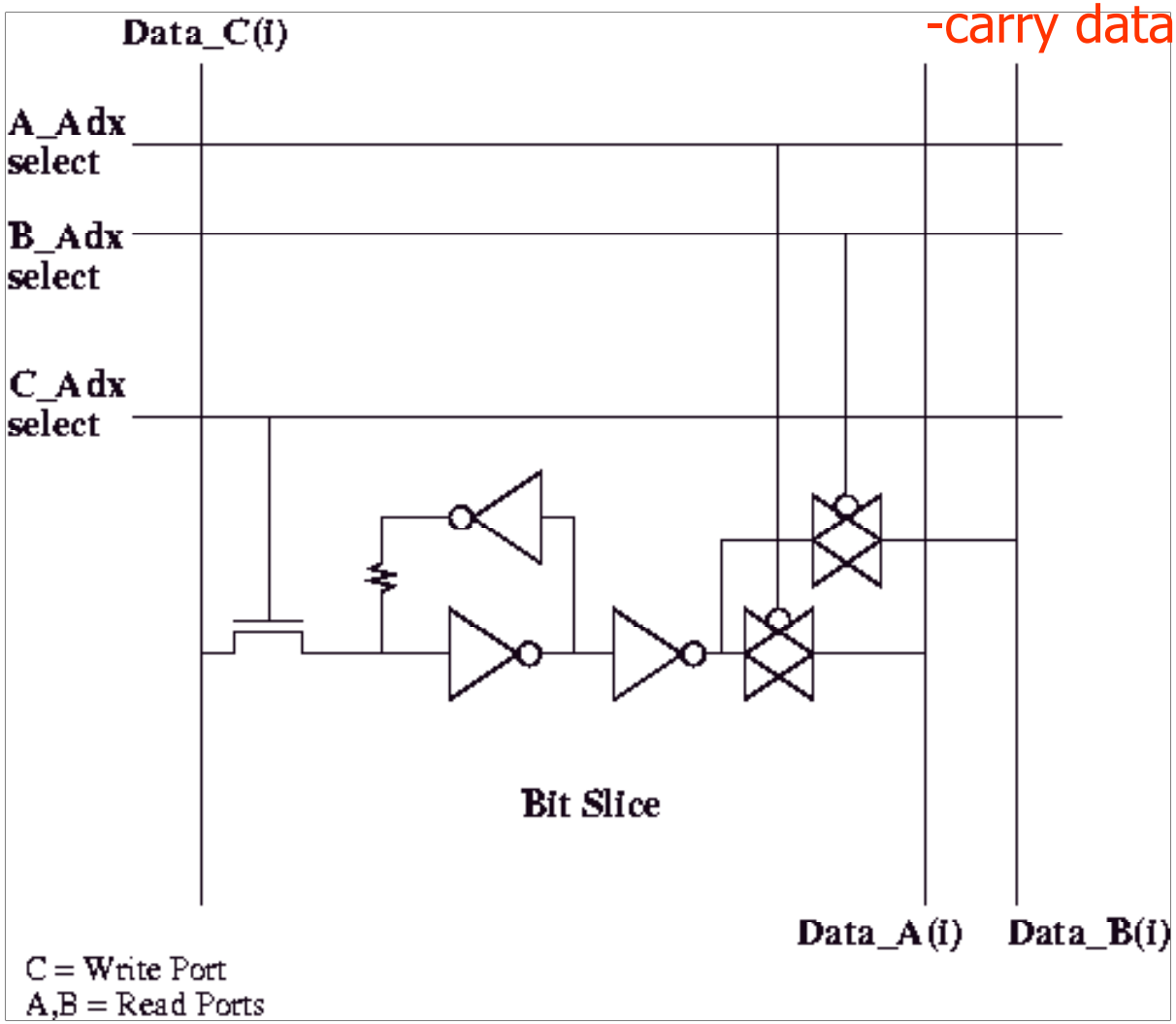
Superscalar Caches

- Increasing issue width => wider caches
- Parallel cache accesses are harder than parallel functional units
- Fundamental difference:
  - Caches have **state**, functional units don't
  - Operation thru one port affects future operations thru others
- Several approaches used
  - True multi-porting
  - Multiple cache copies
  - Virtual multi-porting
  - Multi-banking (interleaving)

True Multiporting of SRAM

"Bit" Lines
-carry data in/out

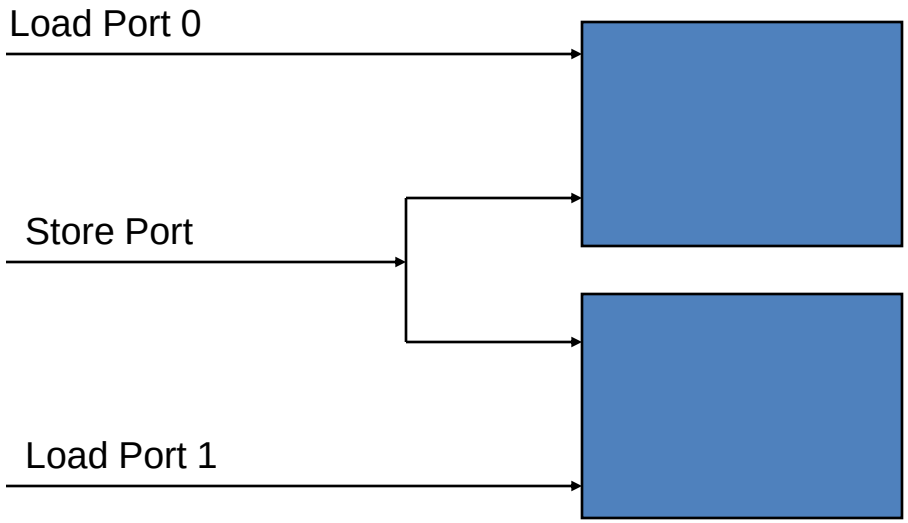
"Word" Lines
-select a row



True Multiporting of SRAM

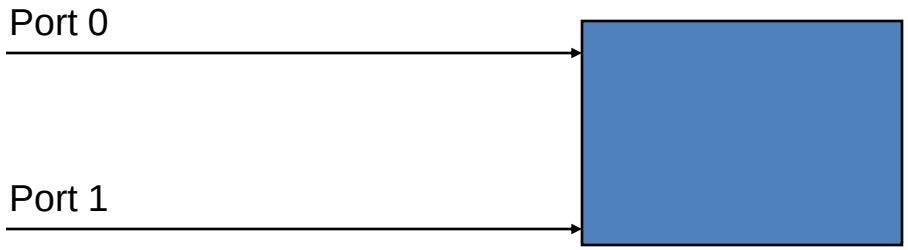
- Would be ideal
- Increases cache area
  - Array becomes wire-dominated
- Slower access
  - Wire delay across larger area
  - Cross-coupling capacitance between wires
- SRAM access difficult to pipeline

Multiple Cache Copies



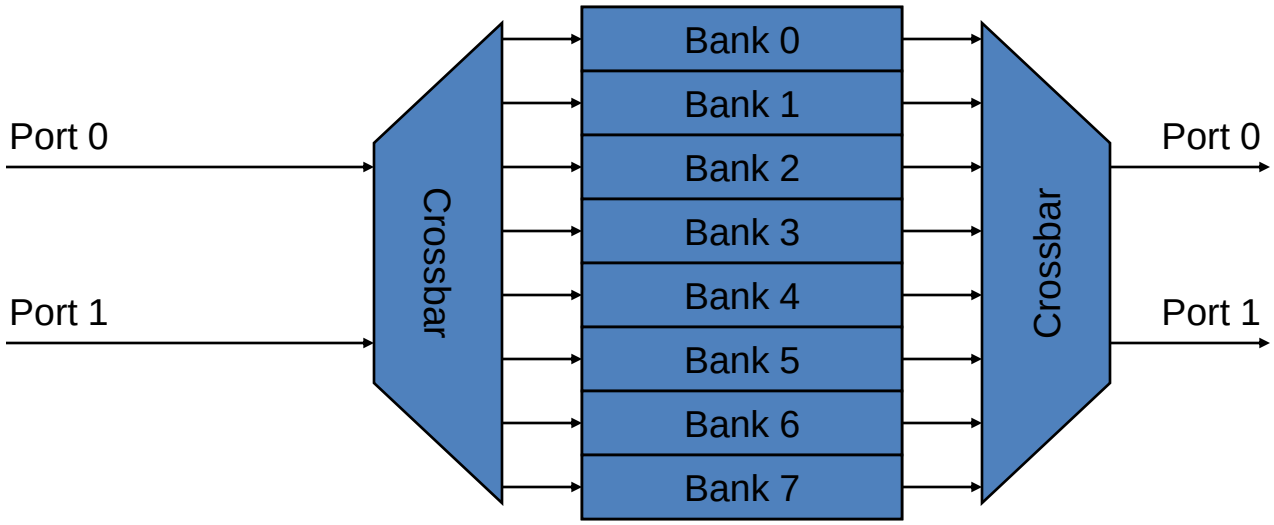
- Used in DEC Alpha 21164, IBM Power4
- Independent load paths
- Single shared store path
  - May be exclusive with loads, or internally dual-ported
- Bottleneck, not practically scalable beyond 2 paths
- Provides some fault-tolerance
  - Parity protection per copy
  - Parity error: restore from known-good copy
  - Avoids more complex ECC (no RMW for subword writes), still provides SEC

Virtual Multiporting



- Used in IBM Power2 and DEC 21264
  - Wave pipelining: pipeline wires WITHOUT latches
- Time-share a single port
- Not scalable beyond 2 ports
- Requires very careful array design to guarantee balanced paths
  - Second access cannot catch up with first access
- Short path constraint limits maximum clock period
- Complicates and reduces benefit of *speed binning*

Multi-banking or Interleaving



- Used in Intel Pentium (8 banks)
- Need routing network
- Must deal with bank conflicts
  - Bank conflicts not known till address generated
  - Difficult in non-data-capture machine with speculative scheduling
    - Replay – looks just like a cache miss
  - Sensitive to bank interleave: fine-grained vs. coarse-grained

Combined Schemes

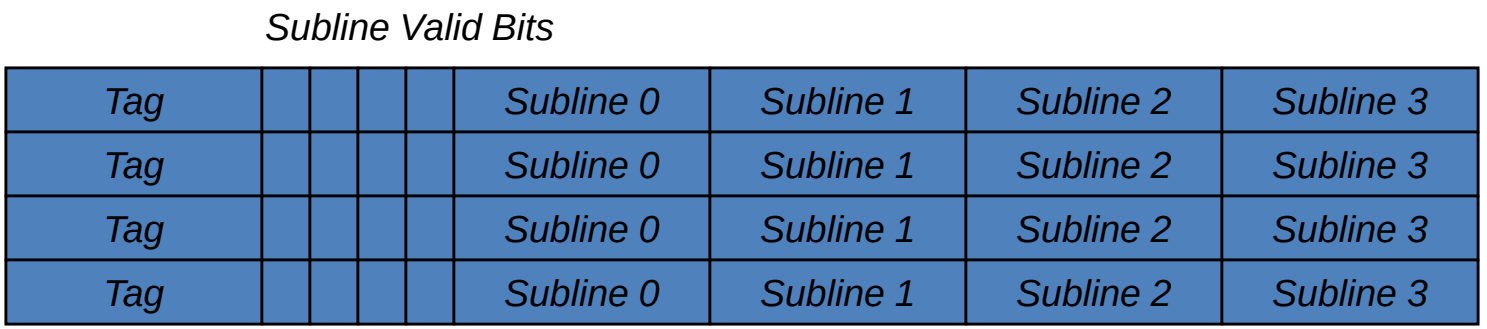
- Multiple banks with multiple ports
- Virtual multiporting of multiple banks
- Multiple ports and virtual multiporting
- Multiple banks with multiply virtually multiported ports
- Complexity!
- No good solution known at this time
  - Current generation superscalars get by with 1-3 ports

Advanced Memory Hierarchy

- Coherent Memory Interface
- Evaluation methods
- Better miss rate: skewed associative caches, victim caches
- Reducing miss costs through software restructuring
- Higher bandwidth: Lock-up free caches, superscalar caches
- **Beyond simple blocks**
- Two level caches

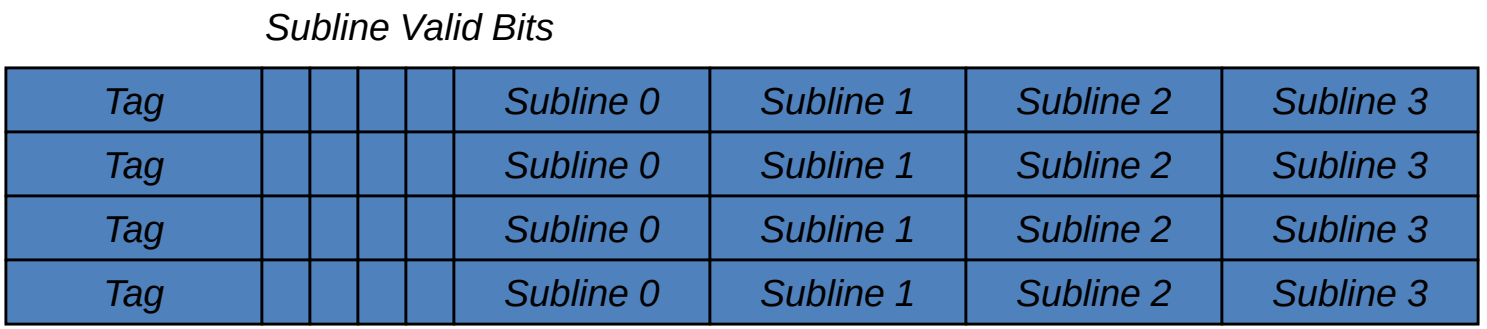
Sublines

- Break blocks into
  - Address block associated with tag
  - Transfer block to/from memory (subline, sub-block)
- Large address blocks
  - Decrease tag overhead
  - But allow fewer blocks to reside in cache (fixed mapping)



Sublines

- Larger transfer block
  - Exploit spatial locality
  - Amortize memory latency
  - But take longer to load
  - Replace more data already cached (more conflicts)
  - Cause unnecessary traffic
- Typically used in large L2/L3 caches to limit tag overhead
- Sublines tracked by MSHR during pending fill



Latency vs. Bandwidth

- Latency can be handled by
  - Hiding (or tolerating) it - out of order issue, nonblocking cache
  - Reducing it – better caches
- Parallelism helps to hide latency
  - MLP – multiple outstanding cache misses overlapped
- But increases bandwidth demand
- Latency ultimately limited by physics

Latency vs. Bandwidth

- Bandwidth can be handled by “spending” more (hardware cost)
  - Wider buses, interfaces
  - Banking/interleaving, multiporting
- Ignoring cost, a well-designed system should never be bandwidth-limited
  - Can’t ignore cost!
- Bandwidth improvement usually increases latency
  - No free lunch
- Hierarchies decrease bandwidth demand to lower levels
  - Serve as traffic filters: a hit in L1 is filtered from L2
- Parallelism puts more demand on bandwidth
- If average b/w demand is not met => infinite queues
  - Bursts are smoothed by queues
- If burst is much larger than average => long queue
  - Eventually increases delay to unacceptable levels

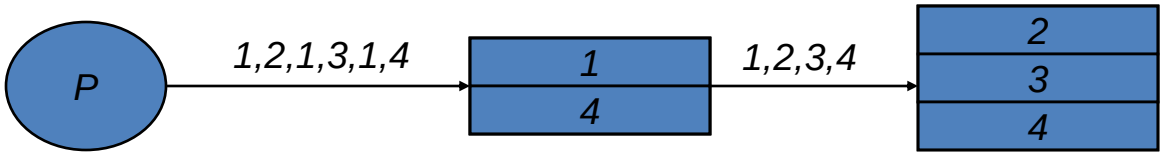
Advanced Memory Hierarchy

- Coherent Memory Interface
- Evaluation methods
- Better miss rate: skewed associative caches, victim caches
- Reducing miss costs through software restructuring
- Higher bandwidth: Lock-up free caches, superscalar caches
- Beyond simple blocks
- **Multilevel caches**

Multilevel Caches

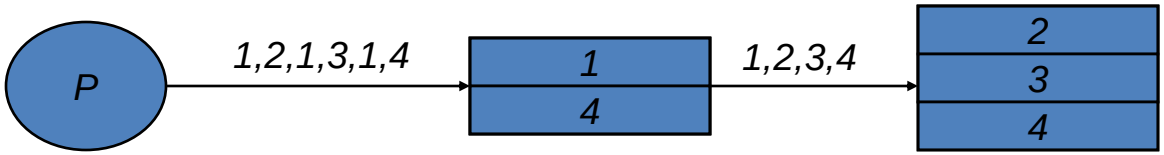
- Ubiquitous in high-performance processors
  - Gap between L1 (core frequency) and main memory too high
  - Level 2 usually on chip, level 3 on or off-chip, level 4 off chip
- Inclusion in multilevel caches
  - Multi-level inclusion holds if L2 cache is superset of L1
  - Can handle virtual address synonyms
  - Filter coherence traffic: if L2 misses, L1 needn't see snoop
  - Makes L1 writes simpler
    - For both write-through and write-back

Multilevel Inclusion



- Example: local LRU not sufficient to guarantee inclusion
  - Assume L1 holds two and L2 holds three blocks
  - Both use local LRU
- Final state: L1 contains 1, L2 does not
  - Inclusion not maintained
- Different block sizes also complicate inclusion

Multilevel Inclusion



- Inclusion takes effort to maintain
  - Make L2 cache have bits or pointers giving L1 contents
  - Invalidate from L1 before replacing from L2
  - In example, removing 1 from L2 also removes it from L1
- Number of pointers per L2 block
  - L2 blocksize/L1 blocksize
- Reading list: [Wang, Baer, Levy ISCA 1989]

Multilevel Miss Rates

- Miss rates of lower level caches
  - Affected by upper level filtering effect
  - LRU becomes LRM, since “use” is “miss”
  - Can affect miss rates, though usually not important
- Miss rates reported as:
  - Miss per instruction
  - Global miss rate
  - Local miss rate
  - “Solo” miss rate
    - L2 cache sees all references (unfiltered by L1)

Advanced Memory Hierarchy

- Coherent Memory Interface
- Evaluation methods
- Better miss rate: skewed associative caches, victim caches
- Reducing miss costs through software restructuring
- Higher bandwidth: Lock-up free caches, superscalar caches
- Beyond simple blocks
- Multilevel caches