Thread-level Parallelism

- Thread-Level parallelism
  - Have multiple program counters
  - Uses MIMD model
  - Targeted for tightly-coupled shared-memory multiprocessors (multicore and multichip)
- For *n* processors, need *n* threads
- Two approaches:
  - Multithreading: interleaves instructions from multiple threads on a single core.
  - Multicore: multiple CPU cores on a single chip (possibly sharing some part of the memory hierarchy)

Multithreading

- Multiple processes can share a processor
- OS handles context switch between processes
- Threads are processes:
  - handled by HW directly
- Can switch threads without requiring OS intervention and a full context switch
- Processor maintains the thread state in HW:
  - Registers
  - Process ID: used to check access on cache and TLB
  - Page Table pointer
  - Program counter
  - Any other unique state

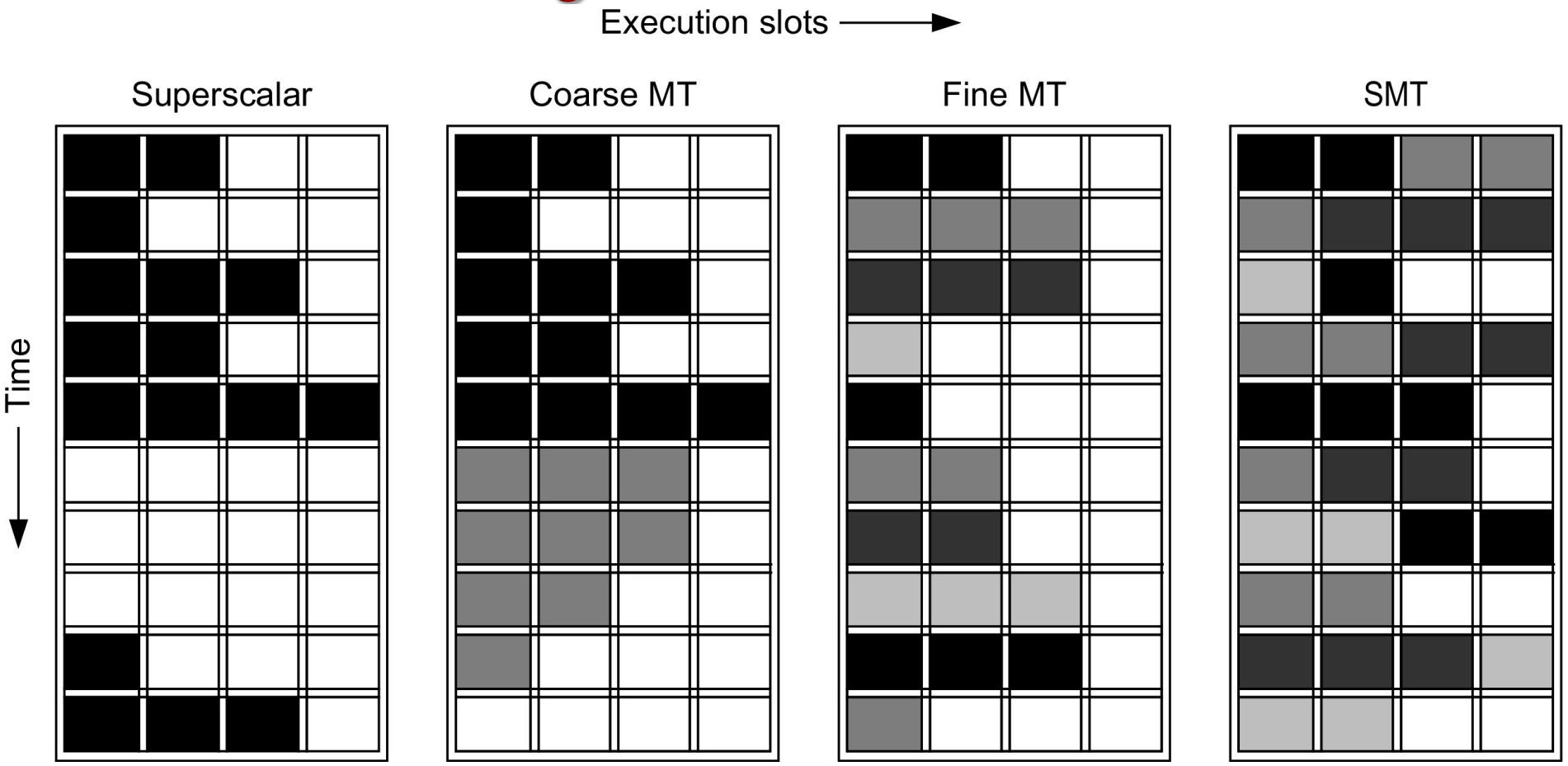
Multithreading: Performance Tradeoff

- Multithreading trades off:
  - Increased throughput: since processor is kept full
    - Versus
  - Maximum single thread performance
    - Latency of completing one thread
- Why?
  - Maximize single thread performance:
    - processor is engaged on that thread or waiting
  - Maximize throughput
    - Keep processor busy

Three Types of Multithreading

1. Coarse grain: switch threads on events that cause stall (like a cache miss)
2. Fine-grained: interleave threads every cycle
3. Simultaneous multithreading: multiple interleaved threads on dynamically-scheduled superscalar allows instructions from different threads to be "executed" together.

Multithreading



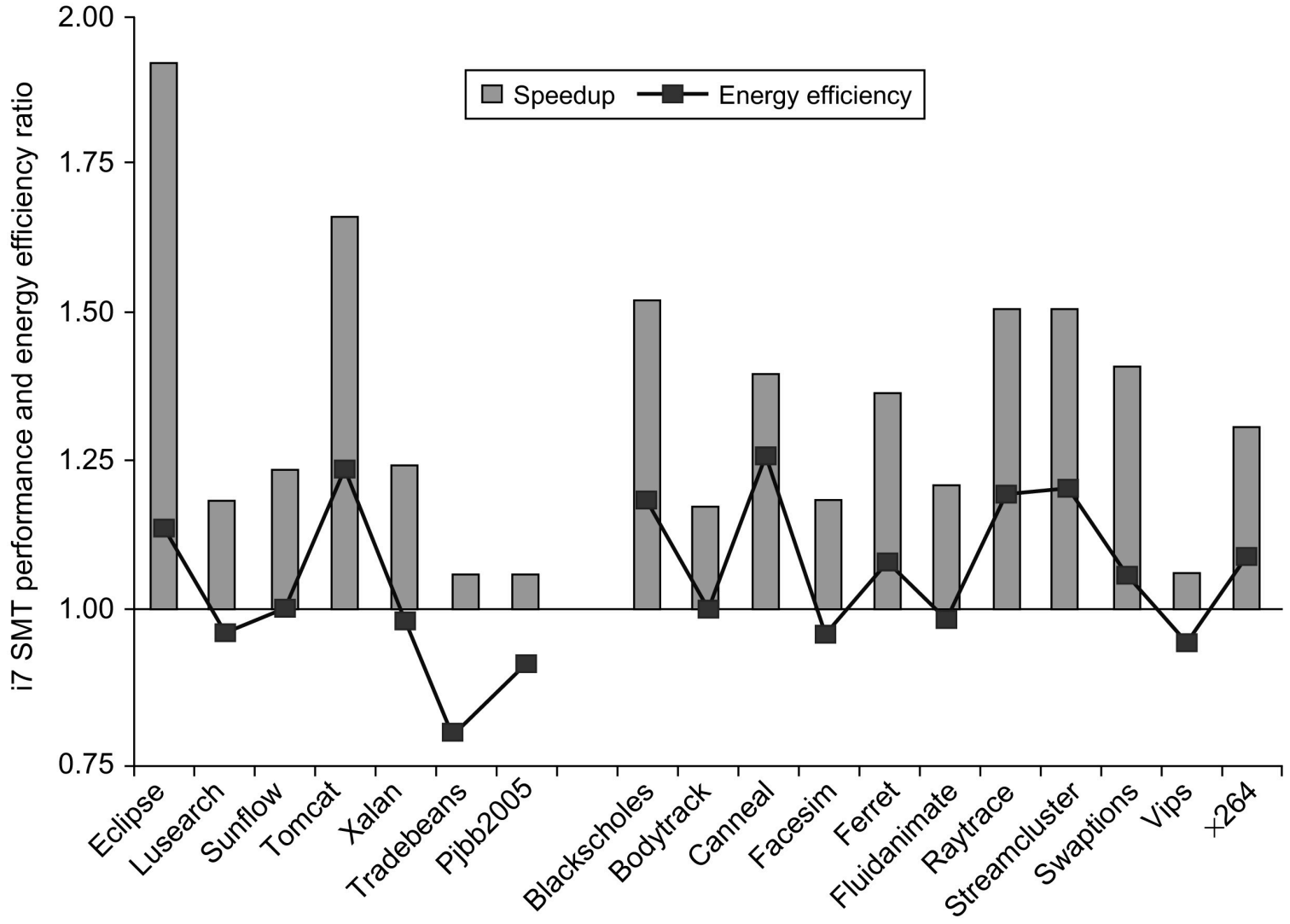
Multithreading Tradeoffs

- Coarse grain:
  - Each thread gets 100% of processor except if stalled
  - With deep pipeline, takes many cycles to get the new thread started
- Fine-grained
  - Good throughput; sacrifices single thread performance
- Simultaneous multithreading
  - Makes effective use of functional units
  - May comprise single thread performance
  - Requires dynamic scheduling HW

Implementing SMT: outline

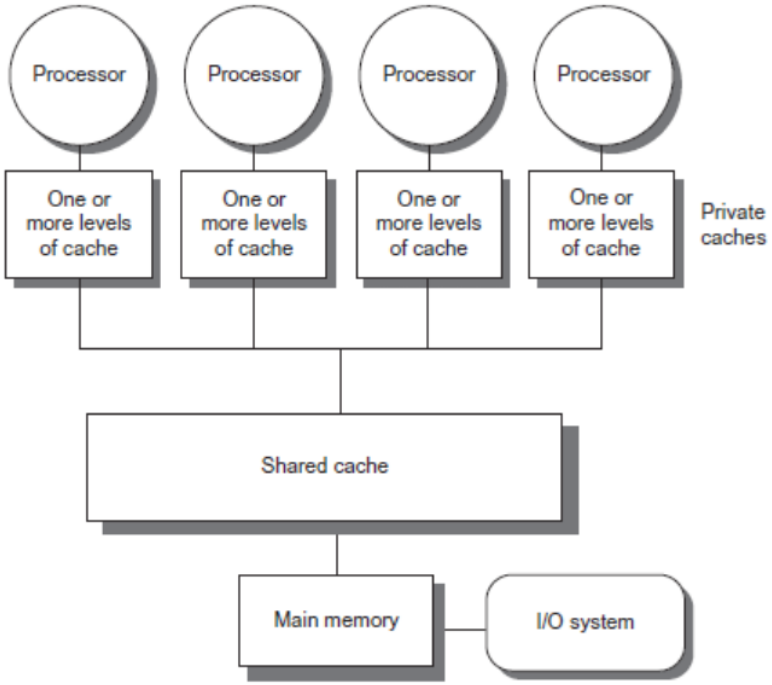
- Use register renaming rather than a reorder buffer and reservation stations.
- Renaming relies on a larger set of physical registers (like the way RSs extend the register set).
- Map updated on commit to the "real" register.
- Renaming map takes logical register names & assigns to large set of physical registers.
- Each active thread has its own renaming map.
- While theoretically we could issue from> 1 thread, too hard in practice and never attempted.
  - Hence "simultaneous" is only in back-end of pipeline

SMT Performance Gain: Intel i7 (one core)



Simple Multicore

- Symmetric multiprocessors (SMP)
  - Small number of cores
  - Share single memory with uniform memory latency
  - Cache coherence through the shared bus going to the LLC.
  - Miss to the shared LLC causes a snoop on the shared bus structure.
  - Bandwidth to the LLC (and hence memory) as well as snooping bandwidth limit scalability.

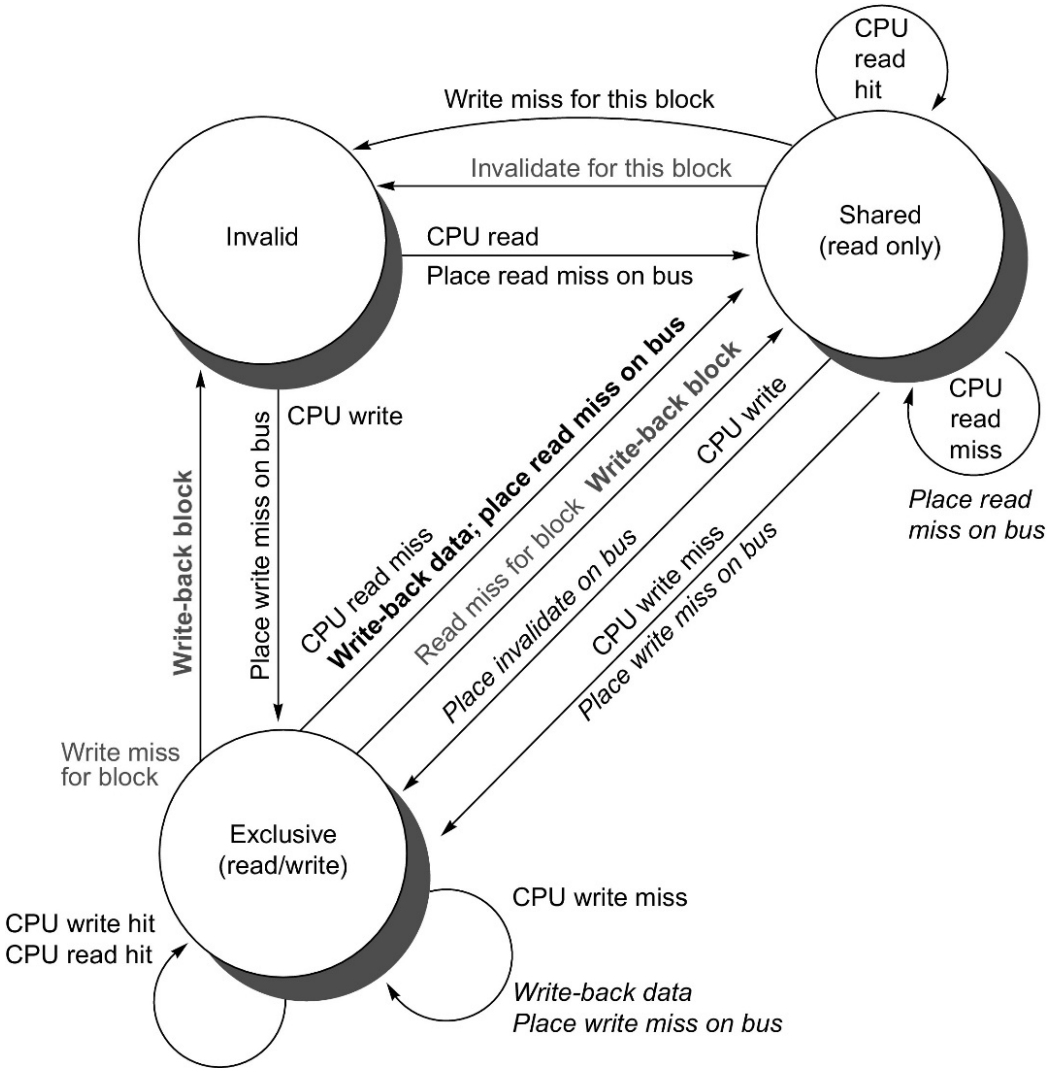


Basic Snooping Algorithm (MSI)

A version of this state machine, runs at every nonshared cache.

Assuming inclusion only the outer level private cache need be checked. If data is found an invalidate is the operation, much check inner cache levels as well.

Shared bus is arbitration point for conflicting accesses.



Modern Multicore

- LLC cache is distributed among the cores
  - NUCA: NonUniform Cache Access
  - Hence also NUMA: Nonuniform Memory Access

Shared LLC is banked: address says which bank caches a block. Inclusion across the entire LLC.

IBM: 8 parallel buses (choose bus by address)

Intel E7: Three rings to connect cores

Since LLC is shared: store coherence information at each block in LLC.

Bit vector says which cores have copies of that cache block.