Prefetching

6.911 Architectures Anonymous

Why Prefetch?

- All about hiding data access latency
  - issue requests for data before they are needed
    - transparent, heuristic look-ahead
    - explicit user-controlled prefetching
    - gray area in-between
  - not without penalties
    - subtleties in shared memory/multiprocessor environments
    - cache pollution
    - higher memory bandwidth => more congestion in multiprocessor networks

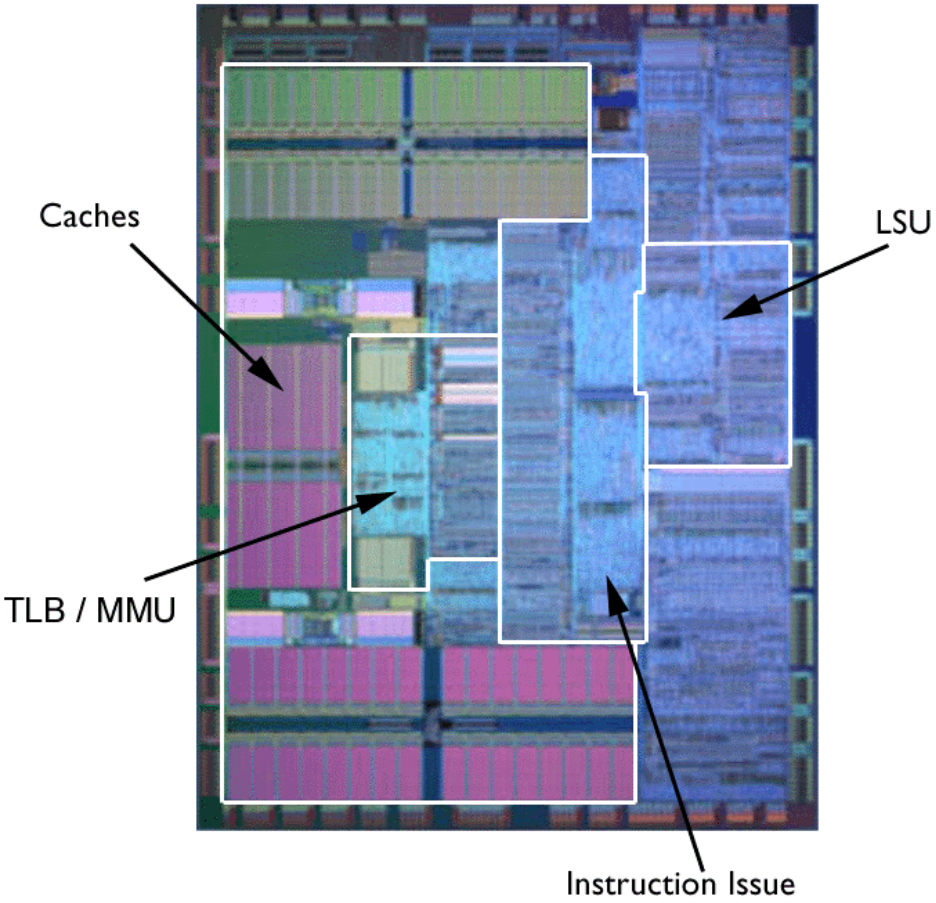
Implementation Options

- Hardware prefetching
  - fast, transparent
  - silicon-inefficient
  - validation difficulties
- Software prefetching
  - slower (but not as slow as a cache miss!)
  - requires some hardware support
  - mistakes are easier to correct

Hardware Implementations

- Some options:
  - Block-based cache replacement
  - Stride-based and markov predictors
  - Instruction trace-based speculative prefetching
- Penalties:
  - validation issues
  - silicon inefficiency
  - application-dependent performance issues

What's Wrong with this Picture?



Software Implementations

- Require some hardware support
  - explicit prefetch instructions
  - slightly modified cache structures
  - hooks for coherency, performance monitoring issues
- Can be explicit in user program, or inferred by compiler (or even by JIT/run-time evaluator!)

Multiprocessor Issues

- Coherency
  - prefetching blocks can lead to cache coherency protocol issues
- Bandwidth considerations

Papers

- Tolerating Latency Though Software Controlled Prefetching…
  - non-binding software controlled prefetching on the DASH architecture
- Two Adaptive Hybrid Cache Coherency...
  - Invalidate/update/adaptive hybrid in multiproc.
- Decoupled Access/Execute Computer Architectures
  - explicit software “prefetching” to an extreme
  - divide program into control and access threads