Brief Introduction to Multiprocessing

more is better?

Multiprocessors

- why would you want a multiprocessor?
- what things can it do well?
- What things can't it do well?
- Multicore vs. big uniprocessor?

What's wrong with the uniprocessor?

- Complexity
- Power
- Lack of Instruction Level Parallelism
- Marginal gains of incremental logic

Uniprocessor Complexity

- The complexity/size of many functional blocks scale quadratically with issue width.
- When IW = 2 or 4, no big deal. Starts to hurt at 8+.
- Rename table has *O x W* ports
  - O = # operands
  - W = fetch width
- Issue queue must do *Q x O x W* comparisons.
  - Q = size of IQ (typically grows as W grows)
- Bypass logic is a W x W interconnect.

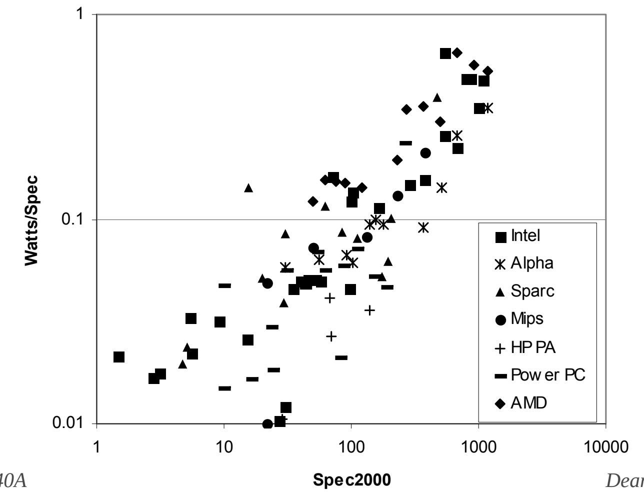
Uniprocessor Complexity

- 4-issue HP PA-8000 – issue queue takes 20% of area.
- [Farkas, et al. 96] claim only 20% gain 4-issue to 8-issue due to cycle time limitations.

The Marginal Utility of Logic

- Example – lines of a 2048-line cache.
- Similarly,
  - Reservation stations
  - Renaming registers
  - Branch predictor size
  - Even issue width
- Exceptions???

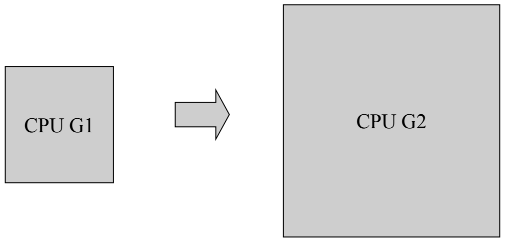
The Price of Performance



Lessons learned

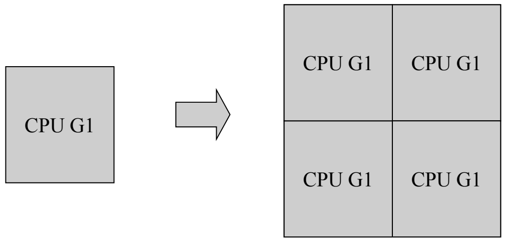
- Marginal utility of each new transistor is decreasing
- If n is the number of transistors
  - Performance is O(sqrt(n))
  - Power and Area are O(n)

Throughput Computing



2X single-thread performance

Throughput Computing



4X throughput

The Alternative

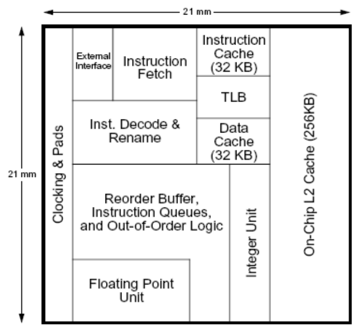


Figure 2. Floorplan for the six-issue dynamic superscalar microprocessor.

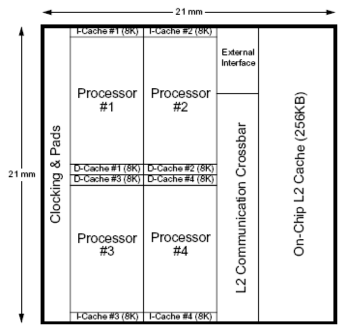


Figure 3. Floorplan for the four-way single-chip multiprocessor.

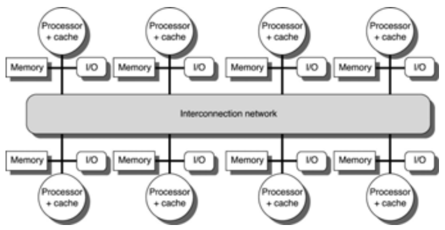
- [Olukotun 1996]

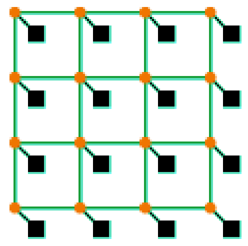
Classifying Multiprocessors

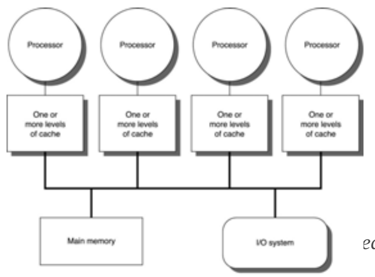
- Interconnection Network
- Memory Topology
- Programming Model

Interconnection Network

- Bus
- Network
- pros/cons?

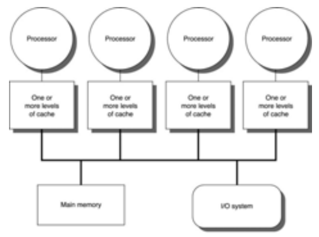


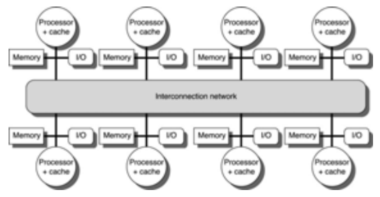


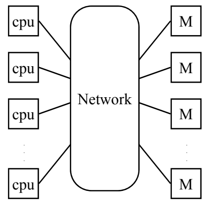


Memory Topology

- UMA (Uniform Memory Access)
- NUMA (Non-uniform Memory Access)
- pros/cons?

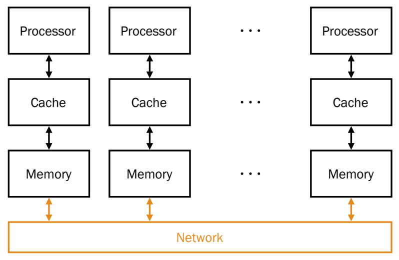






Programming Model

- Shared Memory -- every processor can name every address location
- Message Passing -- each processor can name only it's local memory. Communication is through explicit messages (multicomputer).
- pros/cons?



- *find the max of 100,000 integers on 10 processors.*

Parallel Programming

i = 47

<u>Processor A</u>

index = i++;

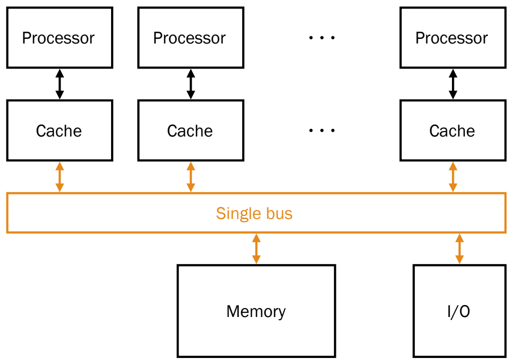
<u>Processor B</u>

index = i++;

- Shared-memory programming requires synchronization to provide mutual exclusion and prevent race conditions
  - locks (semaphores)
  - barriers

Multiprocessor Caches (Shared Memory)

- the problem -- cache coherency
- the solution?

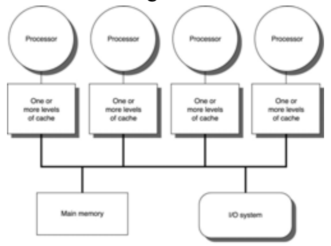


What Does Coherence Mean?

- Informally:
  - Any read must return the most recent write
  - Too strict and very difficult to implement
- Better:
  - A processor sees its own writes to a location in the correct order.
  - Any write must eventually be seen by a read
  - All writes are seen in order ("serialization"). Writes to the same location are seen in the same order by all processors.
- Without these guarantees, synchronization doesn't work.

Cache Coherency

- *write-update*
  - on each write, each cache holding that location updates its value
- *write-invalidate* <= most common
  - on each write, each cache holding that location invalidates the cache line.


Processor
Processor
Processor
Processor
One or more levels of cache
One or more levels of cache
One or more levels of cache
One or more levels of cache
Main memory
I/O system

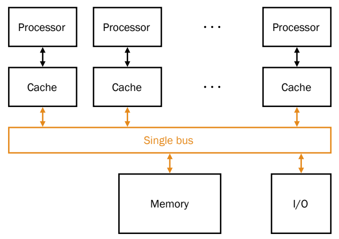
- both schemes MUCH easier on a bus-based multiprocessor
- potentially requires a LOT of messages, but...

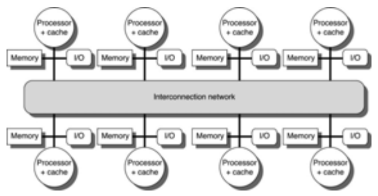
Cache Coherency

- A good cache coherency protocol can avoid sending unnecessary (and expensive) invalidate or update messages.
- Allows each cache line to be in one of several *states*.
- MESI (Illinois)
  - modified
  - exclusive
  - shared
  - invalid

Cache Coherency

- How do you know when an external read/write occurs?
- Snooping protocols
- Directory protocols





Potential Solutions

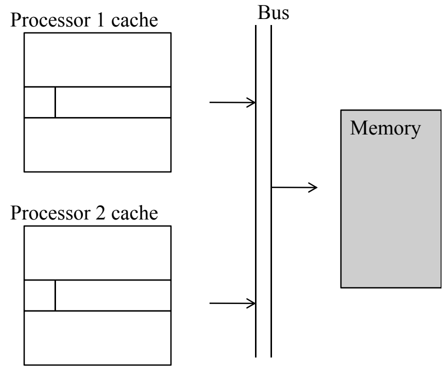
- Snooping Solution (Snoopy Bus):
  - Send all requests for unknown data to all processors
  - Processors snoop to see if they have a copy and respond accordingly
  - Requires “broadcast”, since caching information is at processors
  - Works well with bus (natural broadcast medium)
  - Dominates for small scale machines
- Directory-Based Schemes
  - Keep track of what is being shared in one centralized place
  - Distributed memory => distributed directory (avoids bottlenecks)
  - Send point-to-point requests to processors
  - Scales better than Snoop
  - Actually existed BEFORE Snoop-based schemes

An Example Snoopy Protocol – MESI (or Illinois) protocol

- Invalidation protocol, assumes write-back cache
- Each block of memory is in one state:
  - Clean in all caches and up-to-date in memory
  - Dirty in exactly one cache
  - Not in any caches
- Each cache block is in one state:
  - **(M)**odified: cache has only copy, its writeable, and dirty
  - **(E)**xclusive: cache has only copy, but it’s clean
  - **(S)**hared: block can be read
  - **(I)**nvalid: block contains no data
- Read (and write) misses: cause all caches to snoop
- Writes to shared line are treated as misses

MESI Protocol

Proc 1	Proc 2
lw A	
	lw A
	sw A
	sw A
lw A	
	lw A



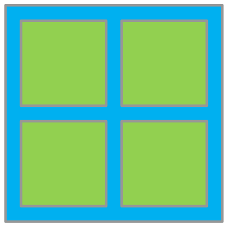
Other protocols

- MESI protocol
  - Big advantage over 3-state protocol (no shared private state) because doesn't require synch messages for private data.
- MOESI = Modified, Owned, Exclusive, Shared, Invalid
  - Owned (dirty in multiple caches, owned in one) => owner responsible for writing back shared, dirty line.
- What traffic does MOESI avoid?

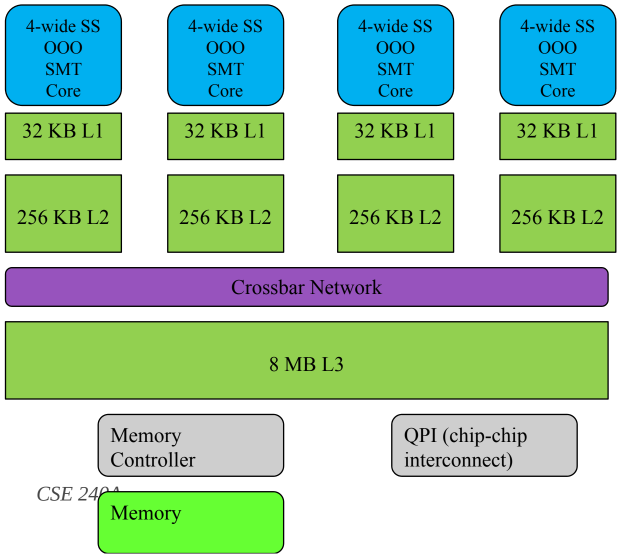
Multicore Architectures

- What is unique/different about multicore architectures?
- Bus or network?
- Shared memory or message passing?
- Need coherence?

Low latency communication. Cores close, memory far away.



A case study – Intel Nehalem (Core i7)



-L1 I cache 4-way, 3 cycles
-L1 D cache 8-way, 4 cycles, physical, write-back, write-allocate
-L2 8-way, non-inclusive, 11 cycles, write-back, write-allocate

L3 16-way inclusive
Directory?
MESI-F (MOESI?)
52 cycles

Single-ISA Heterogeneous Multicore Architectures

- Single-ISA Heterogeneous Multi-Core Architectures: The Potential for Processor Power Reduction, Rakesh Kumar, Keith Farkas, Norm P. Jouppi, Partha Ranganathan, Dean M. Tullsen, In *36th International Symposium on Microarchitecture*, December, 2003.
- If you are putting a bunch of cores on a single processor, why make them all the same?
- Having heterogeneous cores greatly increases the chance that a thread running on the processor finds a core well suited to its execution needs.

Multiprocessors – Key Points

- Network vs. Bus
- Message-passing vs. Shared Memory
- Shared Memory is more intuitive, but creates problems for both the programmer (memory consistency, requiring synchronization) and the architect (cache coherency).