Module 18: "TLP on Chip: HT/SMT and CMP"

Lecture 39: "Simultaneous Multithreading and Chip-multiprocessing"

TLP on Chip: HT/SMT and CMP

- SMT
- Multi-threading
- Problems of SMT
- CMP
- Why CMP?
- Moore's law
- Power consumption?
- Clustered arch.
- ABCs of CMP
- Shared cache design
- Hierarchical MP

Previous Next

Module 18: "TLP on Chip: HT/SMT and CMP"
Lecture 39: "Simultaneous Multithreading and Chip-multiprocessing"

SMT

- Discussed simultaneous multithreading (SMT)
  - Basic goal is to run multiple threads at the same time
  - Helps in hiding large memory latency because even if one thread is blocked due to a cache miss, it is still possible to schedule ready instructions from other threads without taking the overhead of context switch
  - Improves memory level parallelism (MLP)
  - Overall, improves resource utilization enormously as compared to a superscalar processor
  - Latency of a particular thread may not improve, but the overall throughput of the system increases (i.e. average number of retired instructions per cycle)

Multi-threading

- Three design choices for single-core hardware multi-threading
  - Coarse-grain multithreading: Execute one thread at a time; when the running thread is blocked on a long-latency event e.g., cache miss, swap in a new thread; this swap can take place in hardware (needs extra support and extra cycles for flushing the pipe and saving register values unless renamed registers remain pinned)
  - Fine-grain multithreading: Fetch, decode, rename, issue, execute instructions from threads in round robin fashion; improved utilization across cycles, but problem remains within cycle; also if a thread gets blocked on a long-latency event its slots will go wasted for many cycles
  - Simultaneous multithreading (SMT): Mix instructions from all threads every cycle; maximum utilization of resources

Problems of SMT

- Offers a processor that can deliver reasonably good multithreaded performance with fine-grained fast communication through cache
  - Although it is possible to design an SMT processor with small die area increase (5% in Pentium 4), for good performance it is necessary to rethink about resource allocation policies at various stages of the pipe
  - Also, verifying an SMT processor is much harder than the basic underlying superscalar design
  - Must think about various deadlock/livelock possibilities since the threads interact with each other through shared resources on a per-cycle basis
  - Why not exploit the transistors available today to just replicate existing superscalar cores and design a single chip multiprocessor (CMP)?

CMP

- CMP is the mantra of today's microprocessor industry
  - Intel's dual-core Pentium 4: each core is still hyperthreaded (just uses existing cores)
  - Intel's quad-core Whitefield is coming up in a year or so
  - For the server market Intel has announced a dual-core Itanium 2 (code named Montecito); again each core is 2-way threaded
  - AMD has released dual-core Opteron in 2005

- IBM released their first dual-core processor POWER4 circa 2001; next-generation POWER5 also uses two cores but each core is also 2-way threaded
- Sun's UltraSPARC IV (released in early 2004) is a dual-core processor and integrates two UltraSPARC III cores

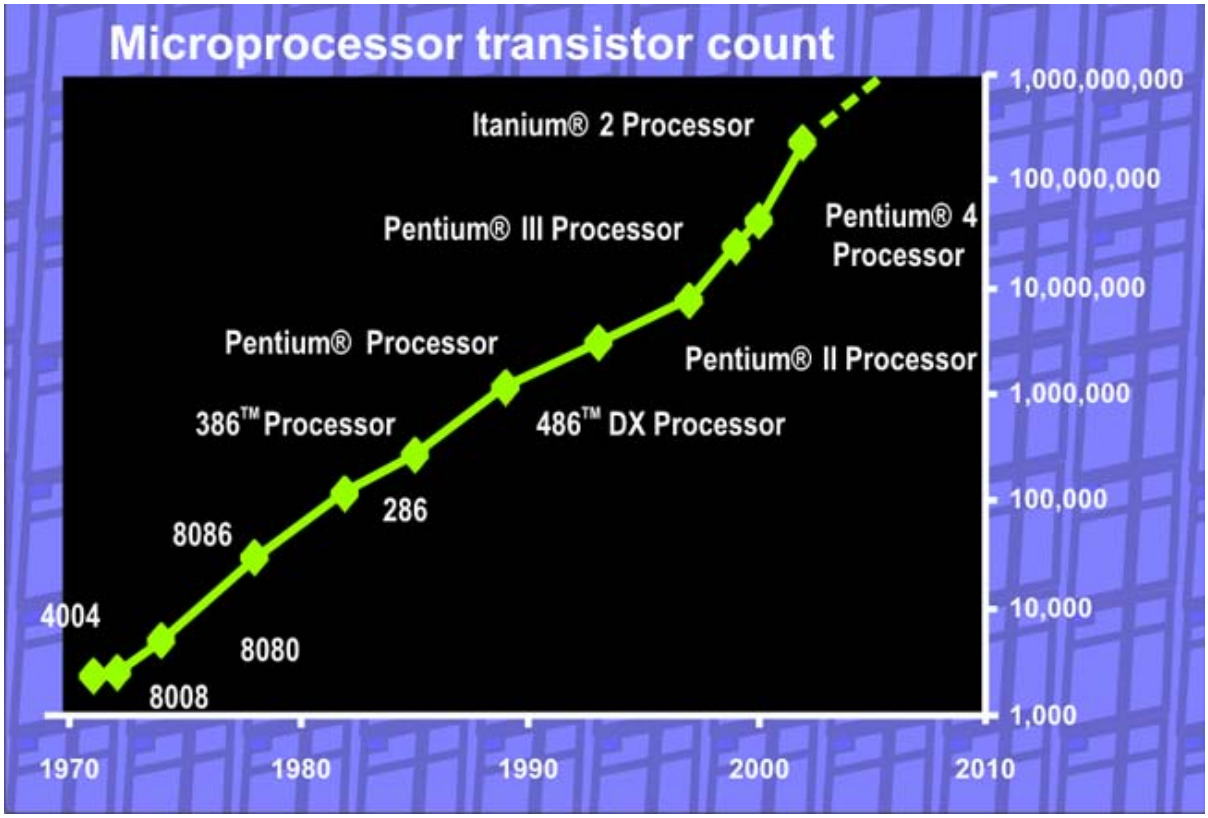
Why CMP?

- Today microprocessor designers can afford to have a lot of transistors on the die
  - Ever-shrinking feature size leads to dense packing
  - What would you do with so many transistors?
  - Can invest some to cache, but beyond a certain point it doesn't help
  - Natural choice was to think about greater level of integration
  - Few chip designers decided to bring the memory and coherence controllers along with the router on the die
  - The next obvious choice was to replicate the entire core; it is fairly simple: just use the existing cores and connect them through a coherent interconnect

Previous Next

Moore's law

- The number of transistors on a die doubles every 18-24 months
  - Exponential growth in available transistor count
  - If transistor utilization is constant, this would lead to exponential performance growth; but life is slightly more complicated
  - Wires don't scale with transistor technology: wire delay becomes the bottleneck
  - Short wires are good: dictates localized logic design
  - But superscalar processors exercise a "centralized" control requiring long wires (or pipelined long wires)
  - However, to utilize the transistors well, we need to overcome the memory wall problem
  - To hide memory latency we need to extract more independent instructions i.e. more ILP
- Extracting more ILP directly requires more available in-flight instructions
  - But for that we need bigger ROB which in turn requires a bigger register file
  - Also we need to have bigger issue queues to be able to find more parallelism
  - None of these structures scale well: main problem is wiring
  - So the best solution to utilize these transistors effectively with a low cost must not require long wires and must be able to leverage existing technology: CMP satisfies these goals exactly (use existing processors and invest transistors to have more of these on-chip instead of trying to scale the existing processor for more ILP)



Power consumption?

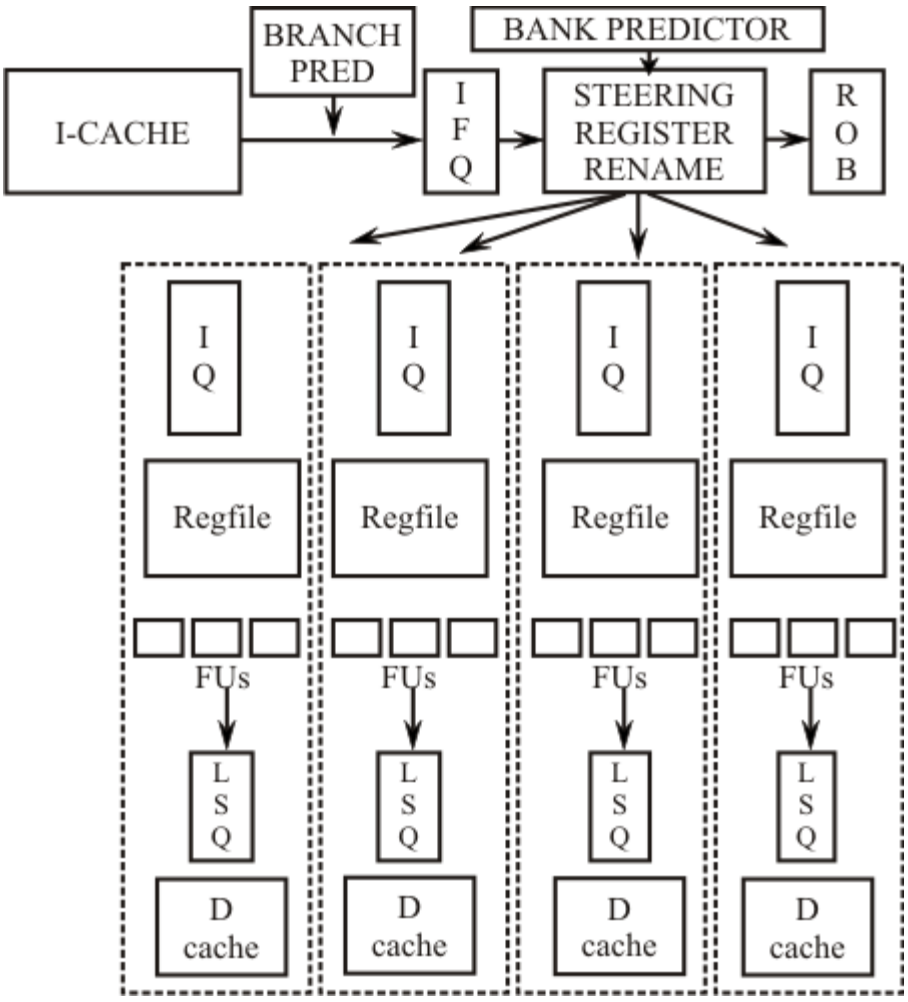
- Hey, didn't I just make my power consumption roughly N-fold by putting N cores on the die?
  - Yes, if you do not scale down voltage or frequency
  - Usually CMPs are clocked at a lower frequency
    - Oops! My games run slower!
  - Voltage scaling happens due to smaller process technology
  - Overall, roughly cubic dependence of power on voltage or frequency
  - Need to talk about different metrics
    - Performance/Watt (same as reciprocal of energy)
    - More general, Performancek+1/Watt (k > 0)
  - Need smarter techniques to further improve these metrics
    - Online voltage/frequency scaling

Previous Next

Module 18: "TLP on Chip: HT/SMT and CMP"
Lecture 39: "Simultaneous Multithreading and Chip-multiprocessing"

Clustered arch.

- An alternative to CMP is clustered microarchitecture
  - Still tries to extract ILP and runs a single thread
  - But divides the execution unit into clusters where each cluster has a separate register file
  - Number of ports per register file goes down dramatically reducing the complexity
  - Can even replicate/partition caches
  - Big disadvantage: keeping the register file and cache partitions coherent; may need global wires
    - Key factor: frequency of communication
  - Also, standard problems of single-threaded execution remain: branch prediction, fetch bandwidth, etc.



May want to steer dependent instructions to the same Cluster to minimize communication

ABCs of CMP

- Where to put the interconnect?

- Do not want to access the interconnect too frequently because these wires are slow
- It probably does not make much sense to have the L1 cache shared among the cores: requires very high bandwidth and may necessitate a redesign of the L1 cache and surrounding load/store unit which we do not want to do; so settle for private L1 caches, one per core
- Makes more sense to share the L2 or L3 caches
- Need a coherence protocol at L2 interface to keep private L1 caches coherent: may use a high-speed custom designed snoopy bus connecting the L1 controllers or may use a simple directory protocol
- An entirely different design choice is not to share the cache hierarchy at all (dual-core AMD and Intel): rids you of the on-chip coherence protocol, but no gain in communication latency

Shared cache design

- Need to be banked
  - How many coherence engines per bank?
  - Notion of home bank? Miss in home bank means what?
  - Snoop or directory?
  - COMA with home bank?

Hierarchical MP

- SMT and CMP add couple more levels in hierarchical multiprocessor design
  - If you just have an SMT processor, among the threads you can do shared memory multiprocessing with possibly the fastest communication; you can connect the SMT processors to build an SMP over a snoopy bus; you can connect these SMP nodes over a network with a directory protocol
  - Can do the same thing with CMP, only difference is that you need to design the on-chip coherence logic (that is not automatically enforced as in SMT)
  - If you have a CMP with each core being an SMT, then you really have a tall hierarchy of shared memory; the communication becomes costlier as you go up the hierarchy; also communication becomes very much non-uniform

Previous Next