ECSE 425 Lecture 25: Multi-threading

H&P Chapter 3

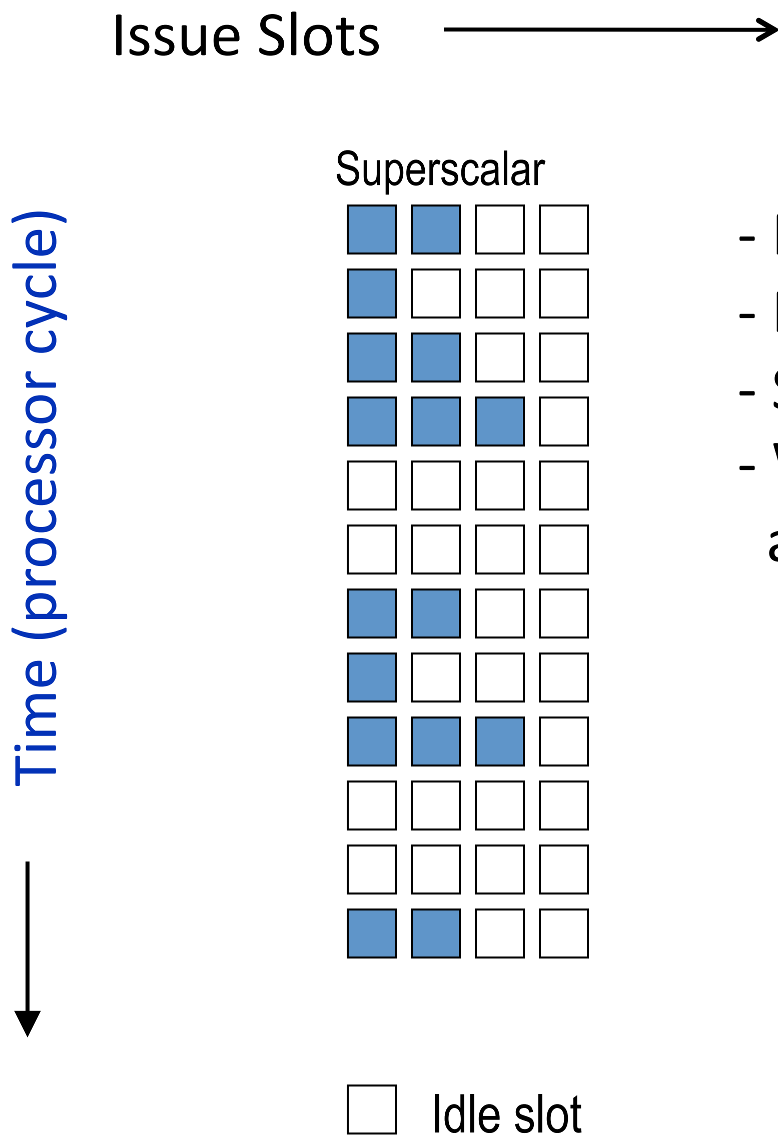
Last Time

- Theoretical and practical limits of ILP
  - Instruction window
  - Branch prediction
  - Register renaming

Today

- Multi-threading
  - Chapter 3.5
- Summary of ILP: Is there a "best" architecture?
  - Chapter 3.6

Single-Threaded Superscalar Machine



- Multiple, dynamic issue
- Multiple FU
- Speculation
- Whole processor stalls when an instruction stalls

Beyond Single Thread ILP

- Parallelism is abundant in many applications
  - Database or scientific codes
  - Medical imaging
- Explicit Parallelism
  - Thread Level Parallelism or Data Level Parallelism
- Thread: process with own instructions and data
  - part of a parallel program of multiple processes,
  - or an independent program
  - Each thread has its own state
- Data Level Parallelism: identical operations on data
  - and lots of data

Thread-Level Parallelism (TLP)

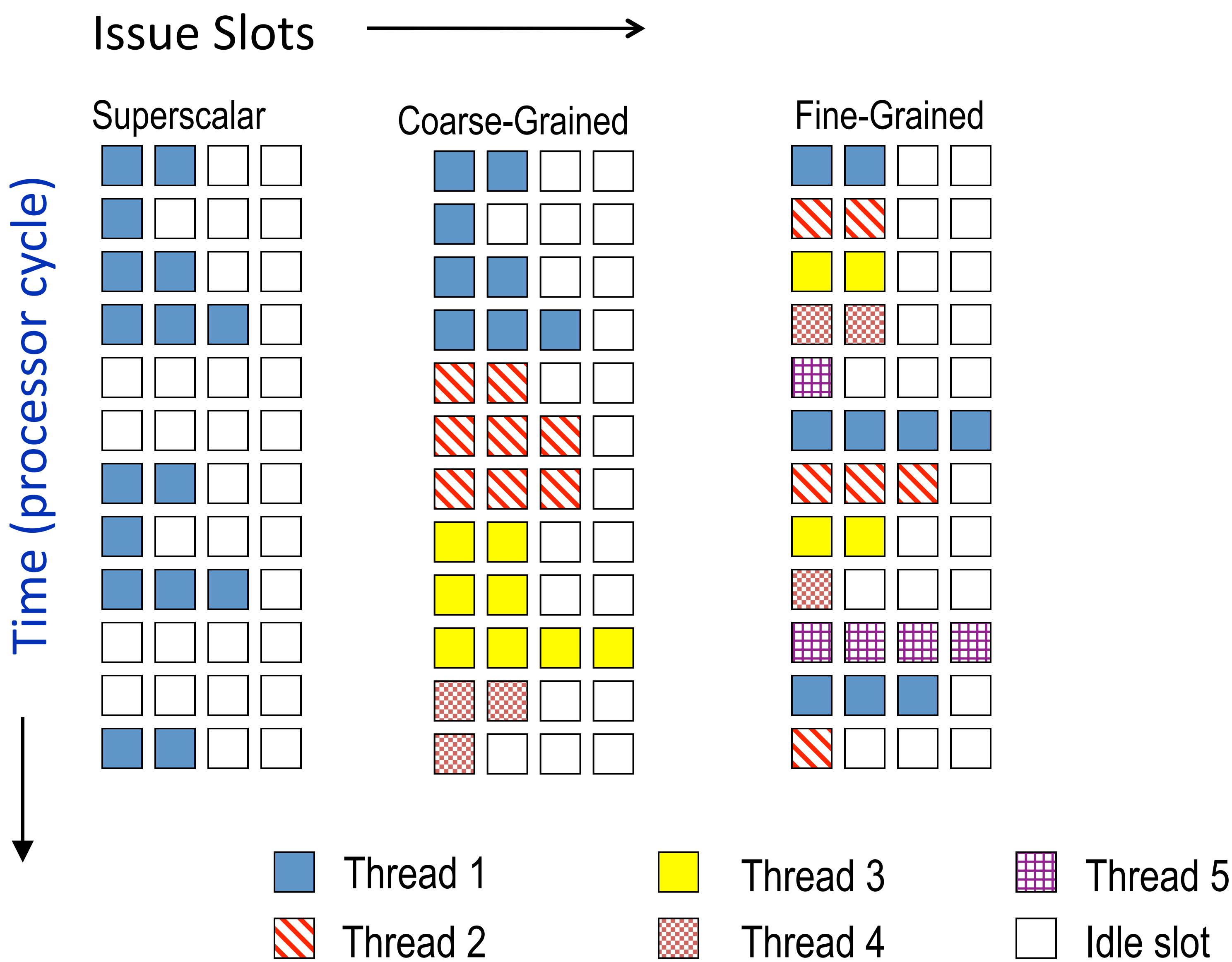
- ILP exploits implicitly parallel operations
  - within a loop or straight-line code segment
- TLP exploits explicit parallelism
  - multiple threads of execution that are inherently parallel
- Goal: Use multiple instruction streams to improve
  1. Throughput of computers that run many programs
  2. Execution time of multi-threaded programs
- TLP could be more cost-effective to exploit than ILP
  - Multithreading allows multiple threads to share the FUs of a single processor in an overlapping fashion
  - Processor must duplicate independent states of threads

Multi-Threading Strategies

Course-Grained Multi-Threading

- Switches threads only on costly stalls, such as L2 misses
- Advantages
  - Fast thread-switching isn't needed
  - Doesn't slow down threads
- Disadvantages
  - Can't hide stalls due to short dependencies
  - Since CPU issues instructions from 1 thread, when a stall occurs, the pipeline must be emptied or frozen
  - New thread must fill pipeline before instructions can complete
- Start-up overhead means coarse-grained multithreading is better for reducing penalty of high cost stalls
  - Pipeline refill << stall time
- Used in IBM AS/400

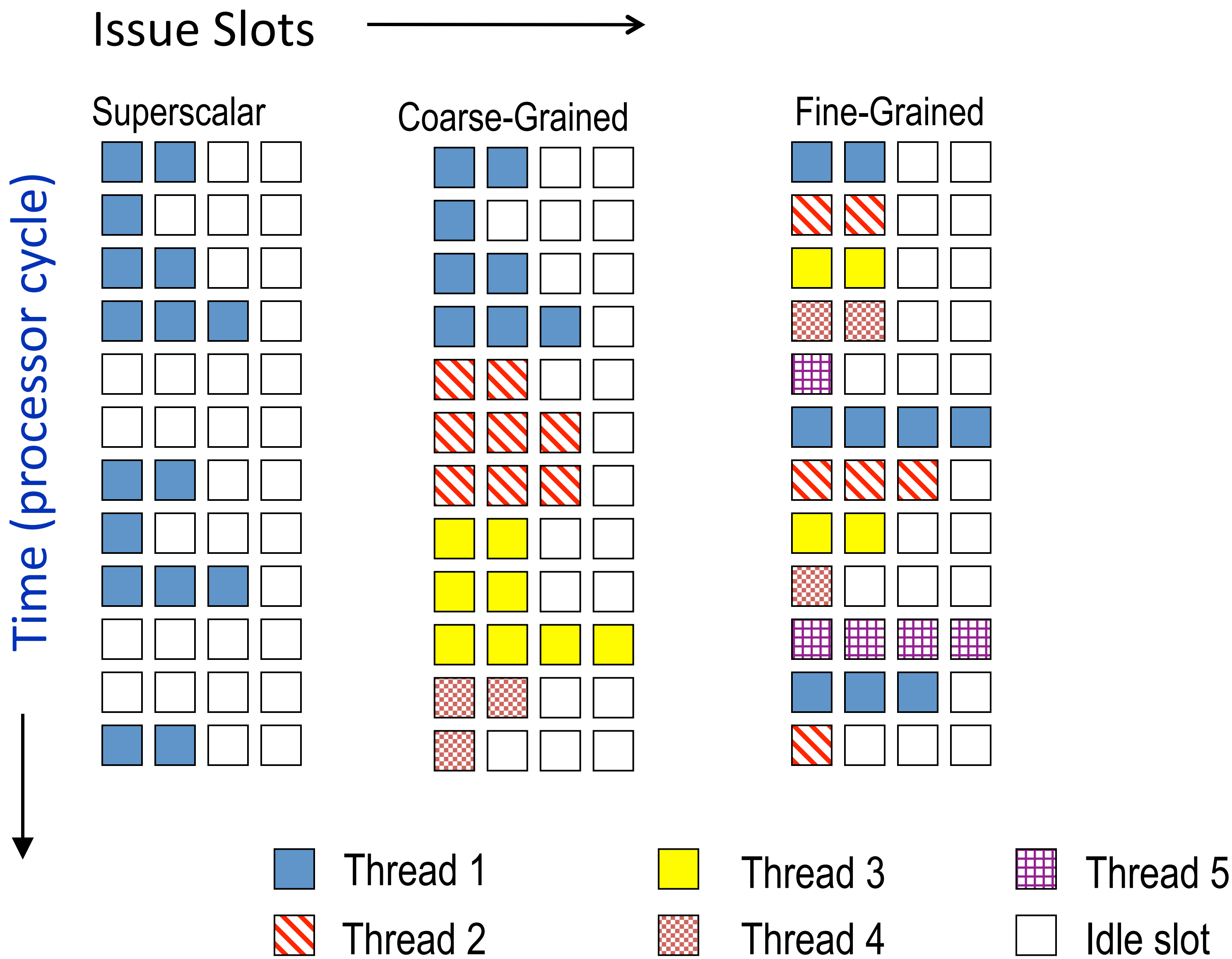
Multi-Threading Strategies



Fine-Grained Multi-Threading

- Switches between threads on each instruction,
  - Causing the execution of multiples threads to be interleaved
  - Usually done in a round-robin fashion, skipping stalled threads
  - CPU must be able to switch threads every clock
- Advantage: can hide both short and long stalls
  - Instructions from other threads execute when a thread stalls
- Disadvantage: slow down execution of individual threads
  - A thread ready to execute without stalls will be delayed by instructions from other threads
- Used in Sun's Niagara

Multi-Threading Strategies



Can We Exploit both ILP and TLP?

- TLP and ILP exploit different parallel structure
- Could a processor designed for ILP exploit TLP?
  - Functional units idle in data path designed for ILP because of either stalls or dependencies in the code
- Could TLP provide independent instructions to keep the processor busy during stalls?
- Could TLP employ the functional units that would otherwise lie idle when insufficient ILP exists?

Simultaneous Multithreading (SMT)

- Simultaneous multithreading (SMT)
  - Leverage multiple-issue and dynamic scheduling
  - Simultaneously exploit both TLP and ILP
- Enough functional units to support multiple threads
- Register renaming and dynamic scheduling
  - Instructions from independent threads can be issued
- Out-of-order execution and completion
  - No inter-thread ordering imposed
- Need to maintain architectural state for each thread
  - Separate PCs, renaming tables, and reorder buffers

Multi-threading Strategies

Single-threaded Performance in SMT

- SMT is a fine-grained multi-threading technique
- Interleaving instructions degrades single-threaded performance
  - Solution: primarily execute instructions from the "preferred thread"
  - "Preferred" designation can rotate
  - But this limits the availability of instructions from other threads when the preferred thread stalls

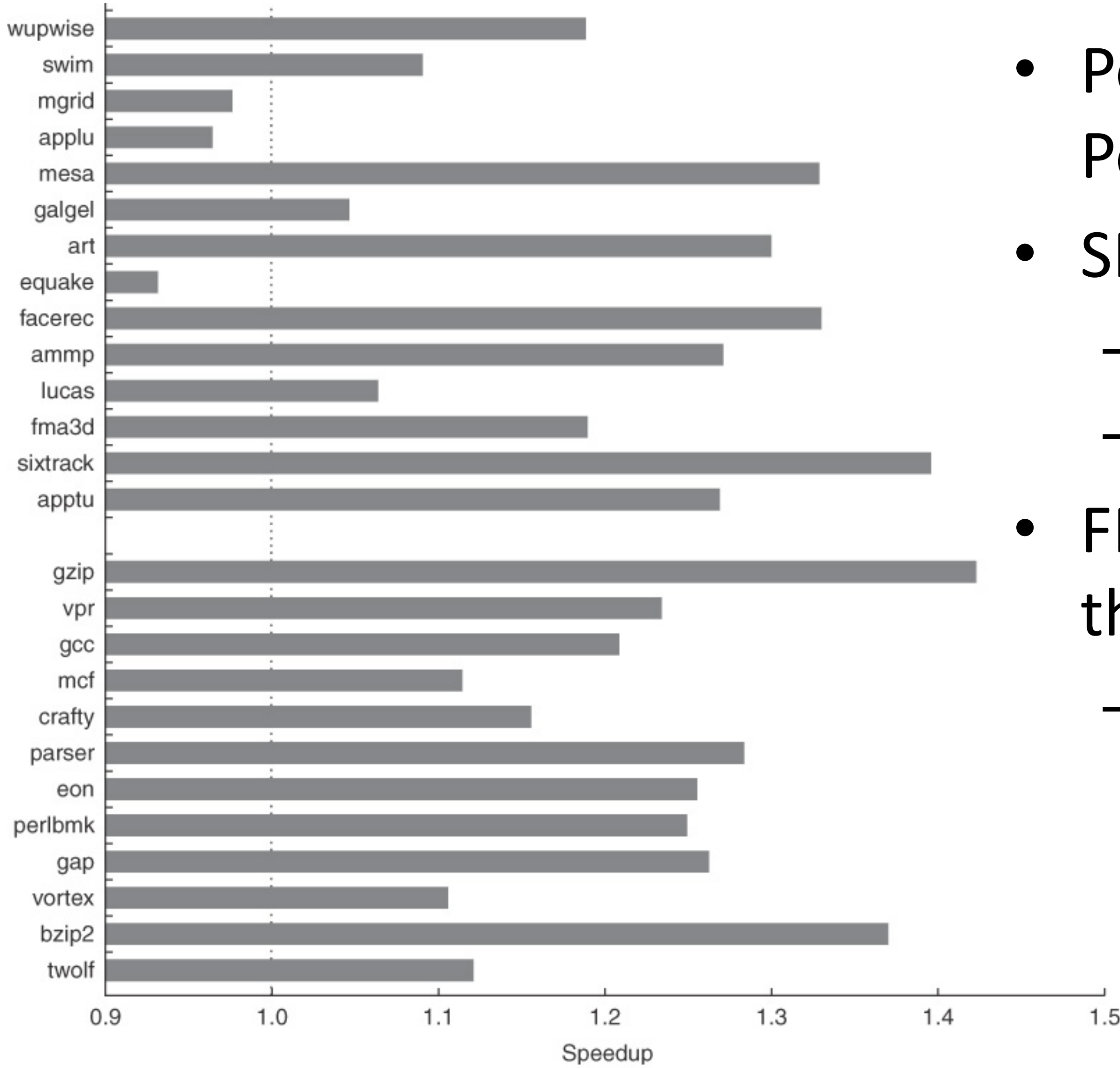
Other Design Challenges in SMT

- Larger register file to hold multiple contexts
- Not increasing clock cycle time, especially in
  - Instruction issue– more candidate instructions need to be considered
  - Instruction completion– choosing which instructions to commit may be challenging
- Cache and TLB conflicts
  - Threads compete for fixed resources
  - Designers must ensure that conflict misses do not degrade performance

SMT Example: IBM Power 5

- More, larger structures, compared with Power 4
- Increased associativity of L1 I$ cache and ITLB
- Per-thread load and store queue
- Larger L2 and L3 cache
- Per-thread instruction prefetch and buffering
- Increased virtual registers from 152 to 240
- Larger issue queues

IBM Power 5 SMT Performance



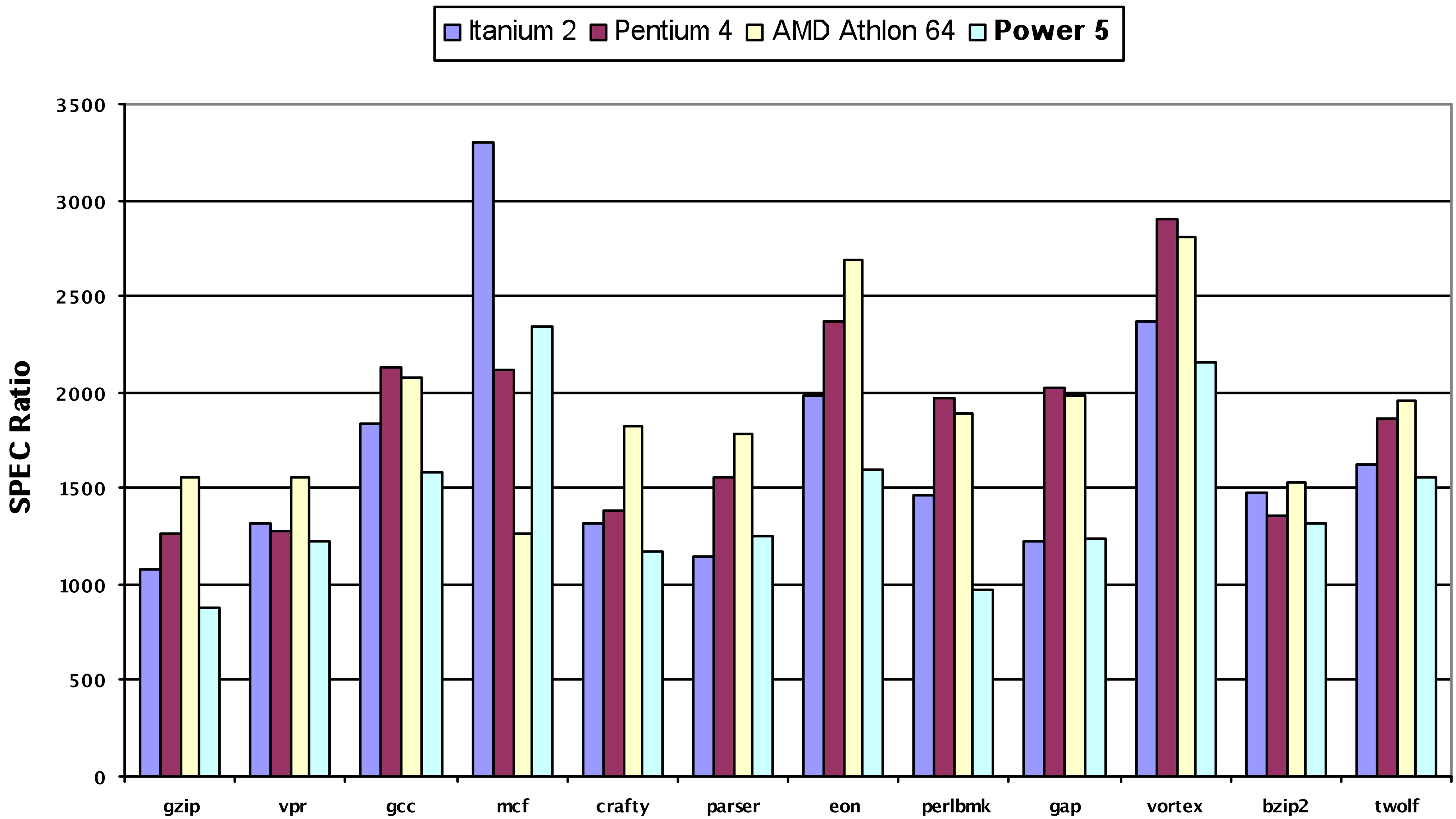
- Power5 w/ SMT vs. Power5 w/o SMT
- SPECRate2000 suite
  - SPECintRate: 1.23×
  - SPECfpRate: 1.16×
- FP applications show the least gains
  - Cache conflicts

Head-to-Head ILP Competition

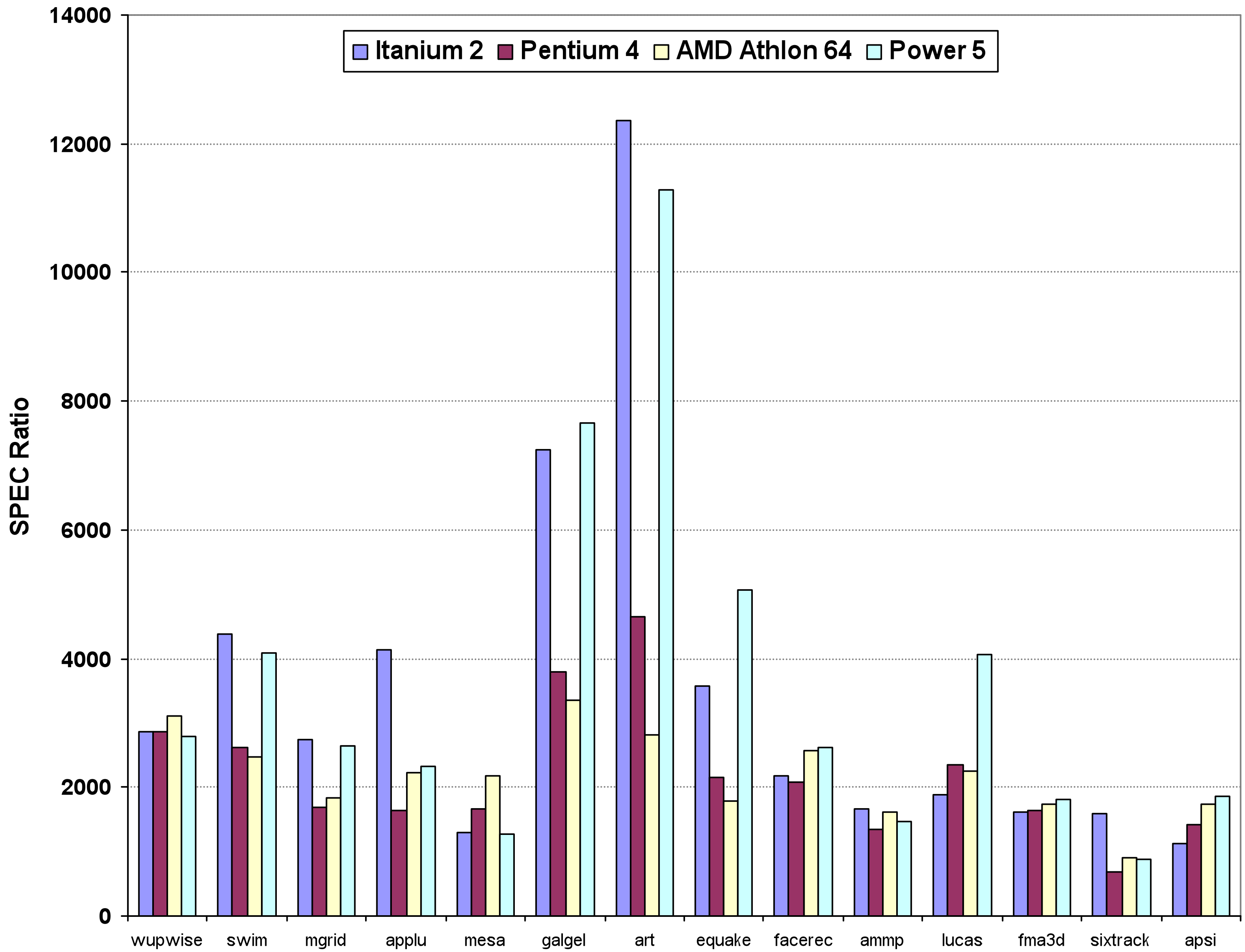
Processor	Micro architecture	Fetch / Issue / Execute	FU	Clock Rate (GHz)	Transis-tors Die size	Power
Intel Pentium 4 Extreme	Speculative dynamically scheduled; deeply pipelined; SMT	3/3/4	7 int. 1 FP	3.8	125 M 122 mm^2	115 W
AMD Athlon 64 FX-57	Speculative dynamically scheduled	3/3/4	6 int. 3 FP	2.8	114 M 115 mm^2	104 W
IBM Power5 (1 CPU only)	Speculative dynamically scheduled; SMT; 2 CPU cores/chip	8/4/8	6 int. 2 FP	1.9	200 M 300 mm^2 (est.)	80W (est.)
Intel Itanium 2	Statically scheduled VLIW-style	6/5/11	9 int. 2 FP	1.6	592 M 423 mm^2	130 W

Which architecture is the best?

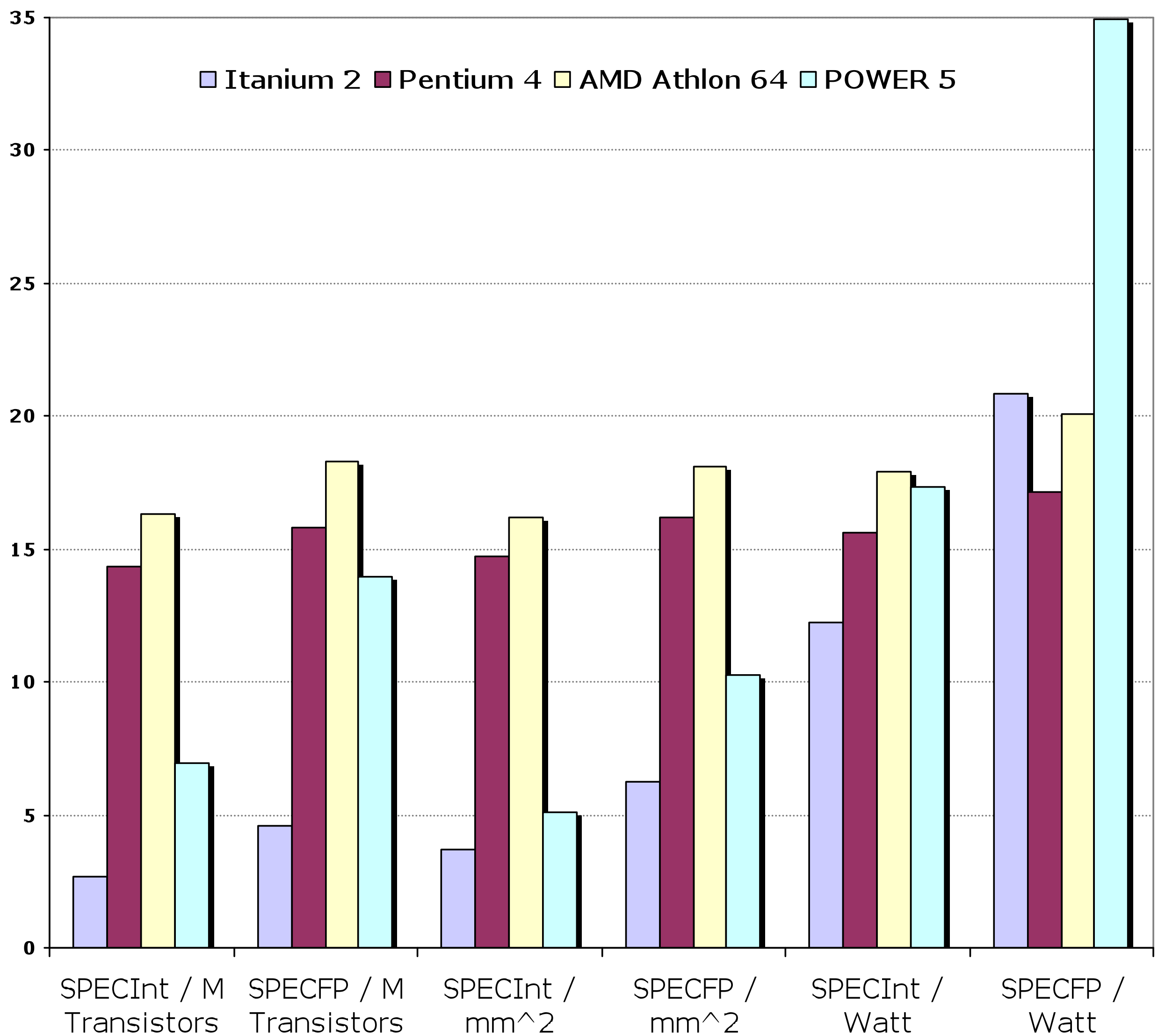
Performance on SPECint2000



Performance on SPECfp2000



Efficiency in Silicon Area and Power



Rank	Itanium2	Pentium 4	AthLon 64	Power 5
Int/Trans	4	2	1	3
FP/Trans	4	2	1	3
Int/area	4	2	1	3
FP/area	4	2	1	3
Int/Watt	4	3	1	2
FP/Watt	2	4	3	1

No Silver Bullet for ILP

- No obvious leader
- SPECInt:
  - The AMD Athlon leads performance
  - Followed by the Pentium 4, Itanium 2, and Power5
- SPECFP:
  - Itanium 2 and Power5 dominate Athlon and P4
- Efficiency
  - Itanium 2 is the least efficient
  - Athlon and P4 are cost-efficient
  - Power5 is the most energy efficient

Additional Limits on ILP

- Doubling issue rates of 3-6 instructions per clock to 6-12 requires that processors be able to
  - issue 3 or 4 data memory accesses per cycle,
  - resolve 2 or 3 branches per cycle,
  - rename and access more than 20 registers per cycle, and
  - fetch 12 to 24 instructions per cycle.
- Implementing this likely means sacrificing clock rate
  - E.g. Among the four processors, Itanium 2 (VLIW) has
    - widest issue processor
    - slowest clock rate
    - and consumes the most power!

Additional Limits on ILP, Cont'd

- Modern processors are mostly power limited
  - Increasing performance also increases power
  - Energy efficiency: does performance grow faster than power?
- Multiple-issue techniques are energy inefficient
  - Logic overhead grows faster than the issue rate
  - Growing gap between peak and sustained issue rates
    - Performance grows with the sustained performance
    - Power grows with the peak performance
  - Speculation is inherently inefficient

Summary

- The power inefficiency and complexity of exploiting ILP seem to limit CPUs to 3-6 wide issue
- Exploiting explicit parallelism is the next step
  - Data-level parallelism or
  - Thread-level parallelism
- Coarse vs. Fine grained multi-threading
  - Switching on big stalls vs. every clock cycle
  - Simultaneous Multi-threading is fine grained multithreading that exploits both ILP and TLP
- The application-specific balance of ILP and TLP is decided by the marketplace

Next Time

- Multiprocessors and Thread-level Parallelism
  - Read Chapter 4.1!