高等计算机系统结构

多线程技术

（第七讲）

程 旭

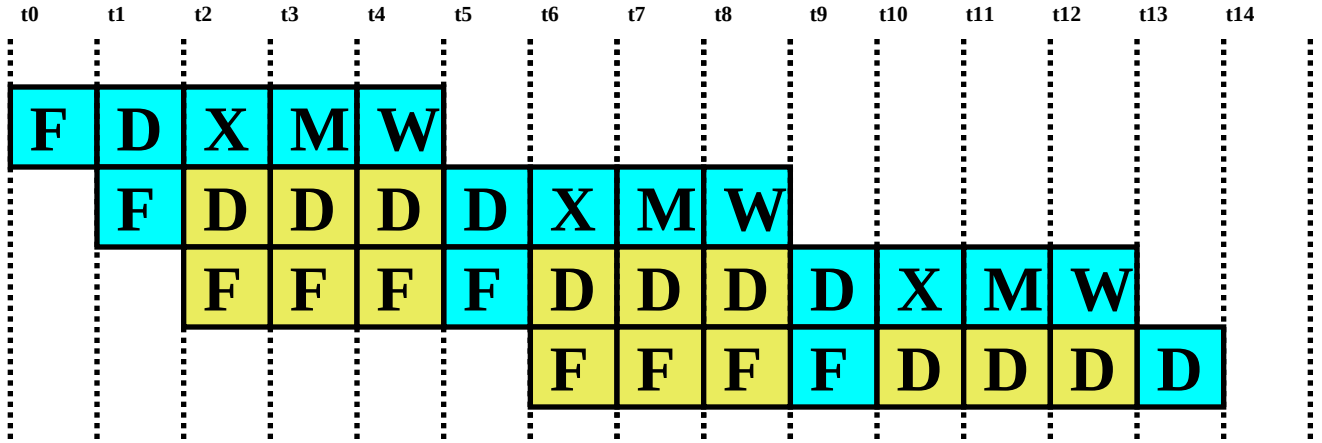
2014年5月5日

多线程处理（Multithreading）

- 从单一顺序控制线程中继续抽取指令级并行越来越困难。
- 许多工作负载可以利用线程级并行（thread-level parallelism：TLP)
  - TLP：来自多到程序（运行独立的多个顺序任务）
  - TLP：来自多线程（通过并行执行线程来加速单个任务）
- 多线程处理技术（Multithreading）利用TLP来改进单个处理器的效用（utilization

流水线冒险

LW r1, 0(r2)
LW r5, 12(r1)
ADDI r5, r5, #12
SW 12(r1), r5



- 每条指令都可能与其他指令相关

What is usually done to cope with this?

- *interlocks (slow)*
- *or bypassing (needs hardware, doesn't help all hazards)*

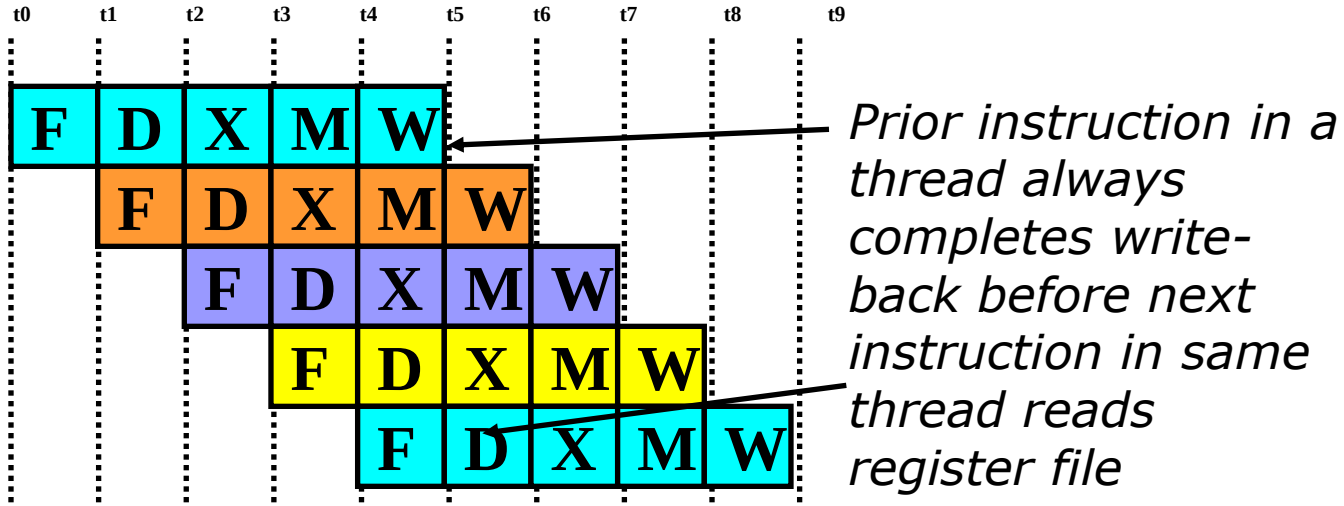
多线程处理

如何确保流水线中的指令之间没有相关?

-- 方法之一：在同一流水线上，从不同程序线程选取指令，交叉执行（interleave execution）

Interleave 4 threads, T1-T4, on non-bypassed 5-stage pipe

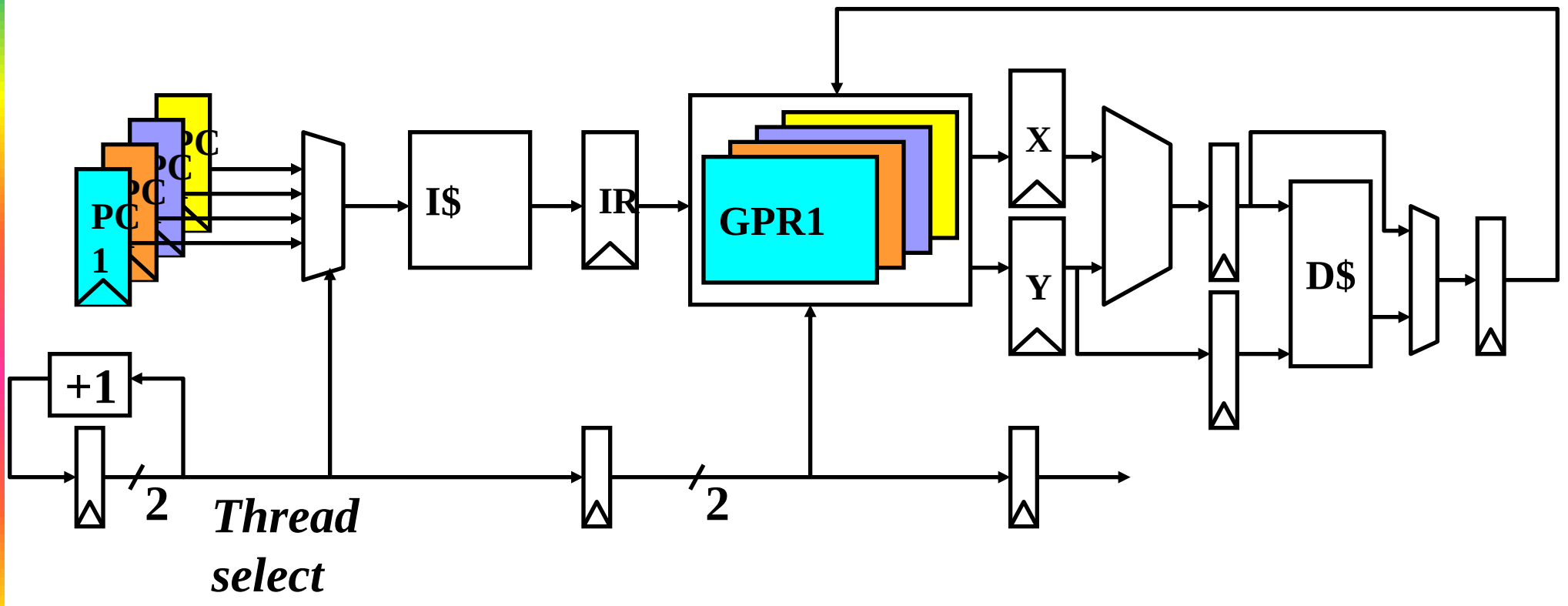
T1: LW r1, 0(r2)
T2: ADD r7, r1, r4
T3: XORI r5, r4, #12
T4: SW 0(r7), r5
T1: LW r5, 12(r1)



CDC 6600 外设处理机(Cray, 1964)

- First multithreaded hardware
- 10 “virtual” I/O processors
- Fixed interleave on simple pipeline
- Pipeline has 100ns cycle time
- Each virtual processor executes one instruction every 1000ns
- Accumulator-based instruction set to reduce processor state

简易多线程流水线



- Have to carry thread select down pipeline to ensure correct state bits read/written at each pipe stage
- Appears to software (including OS) as multiple, albeit slower, CPUs

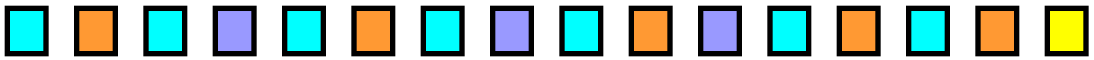
多线程的开销

- 每个线程都需要自己的用户状态
  - PC
  - GPRs
- 还需要自己的系统状态
  - Virtual-memory page-table-base register
  - Exception-handling registers
- *其他开销:*
  - Additional cache/TLB conflicts from competing threads
  - (or add larger cache/TLB capacity)
  - More OS overhead to schedule more threads (where do all these threads come from?)

线程调度策略

- Fixed interleave *(CDC 6600 PPUs, 1964)*
  - Each of N threads executes one instruction every N cycles
  - If thread not ready to go in its slot, insert pipeline bubble

- Software-controlled interleave *(TI ASC PPUs, 1971)*
  - OS allocates S pipeline slots amongst N threads
  - Hardware performs fixed interleave over S slots, executing whichever thread is in that slot



- Hardware-controlled thread scheduling *(HEP, 1982)*
  - Hardware keeps track of which threads are ready to go
  - Picks next thread to execute based on hardware priority scheme

Denelcor HEP
(Burton Smith, 1982)

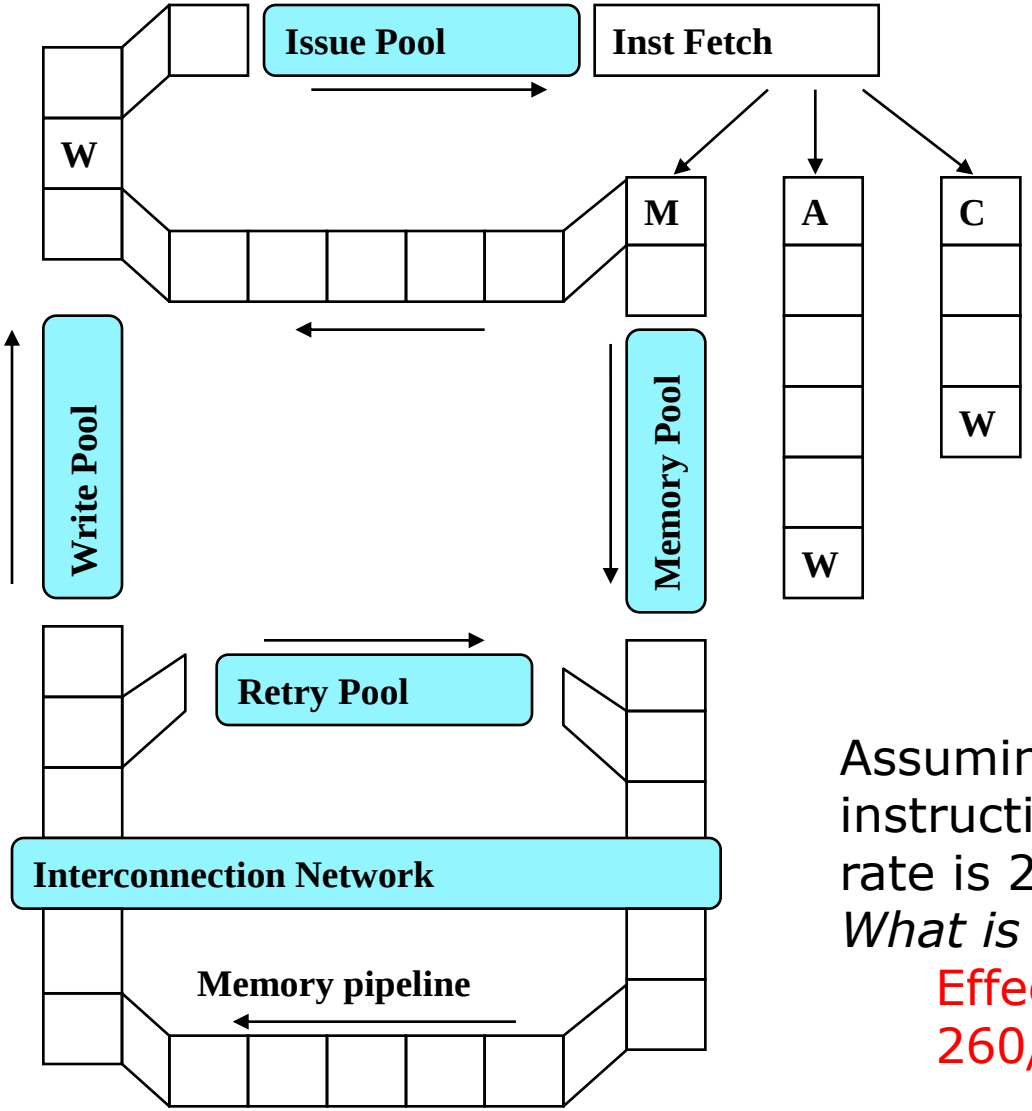
First commercial machine to use hardware threading in main CPU

- 120 threads per processor
- 10 MHz clock rate
- Up to 8 processors
- precursor to Tera MTA (Multithreaded Architecture)

Tera MTA (1990-)

- Up to 256 processors
- Up to 128 active threads per processor
- Processors and memory modules populate a sparse 3D torus interconnection fabric
- Flat, shared main memory
  - No data cache
  - Sustains one main memory access per cycle per processor
- GaAs logic in prototype, 1KW/processor @ 260MHz
  - Second version CMOS, MTA-2, 50W/processor
  - New version, XMT, fits into AMD Opteron socket, runs at 500MHz

MTA Pipeline



- Every cycle, one VLIW instruction from one active thread is launched into pipeline
- Instruction pipeline is 21 cycles long
- Memory operations incur ~150 cycles of latency

Assuming a single thread issues one instruction every 21 cycles, and clock rate is 260 MHz…

What is single-thread performance?

Effective single-thread issue rate is 260/21 = 12.4 MIPS

Coarse-Grain Multithreading

- Tera MTA designed for supercomputing applications with large data sets and low locality
  - No data cache
  - Many parallel threads needed to hide large memory latency
- Other applications are more cache friendly
  - Few pipeline bubbles if cache mostly has hits
  - Just add a few threads to hide occasional cache miss latencies
  - Swap threads on cache misses

IBM PowerPC RS64-IV (2000)

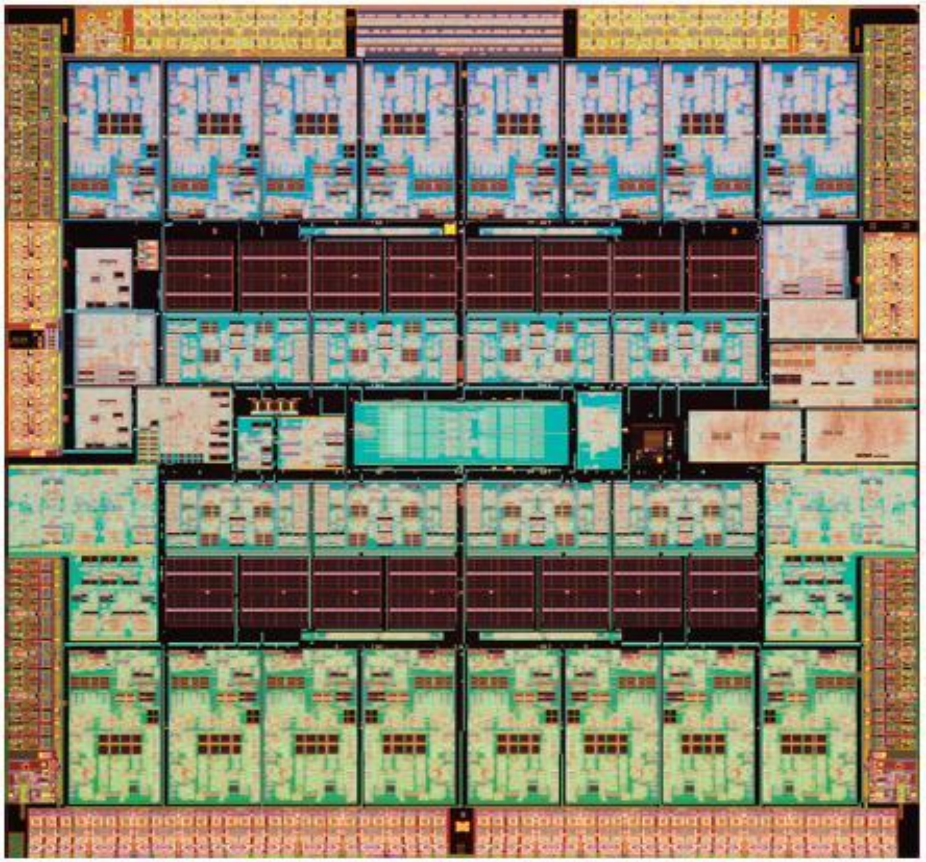
- Commercial coarse-grain multithreading CPU
- Based on PowerPC with quad-issue in-order five-stage pipeline
- Each physical CPU supports two virtual CPUs
- On L2 cache miss, pipeline is flushed and execution switches to second thread
  - short pipeline minimizes flush penalty (4 cycles), small compared to memory access latency
  - flush pipeline to simplify exception handling

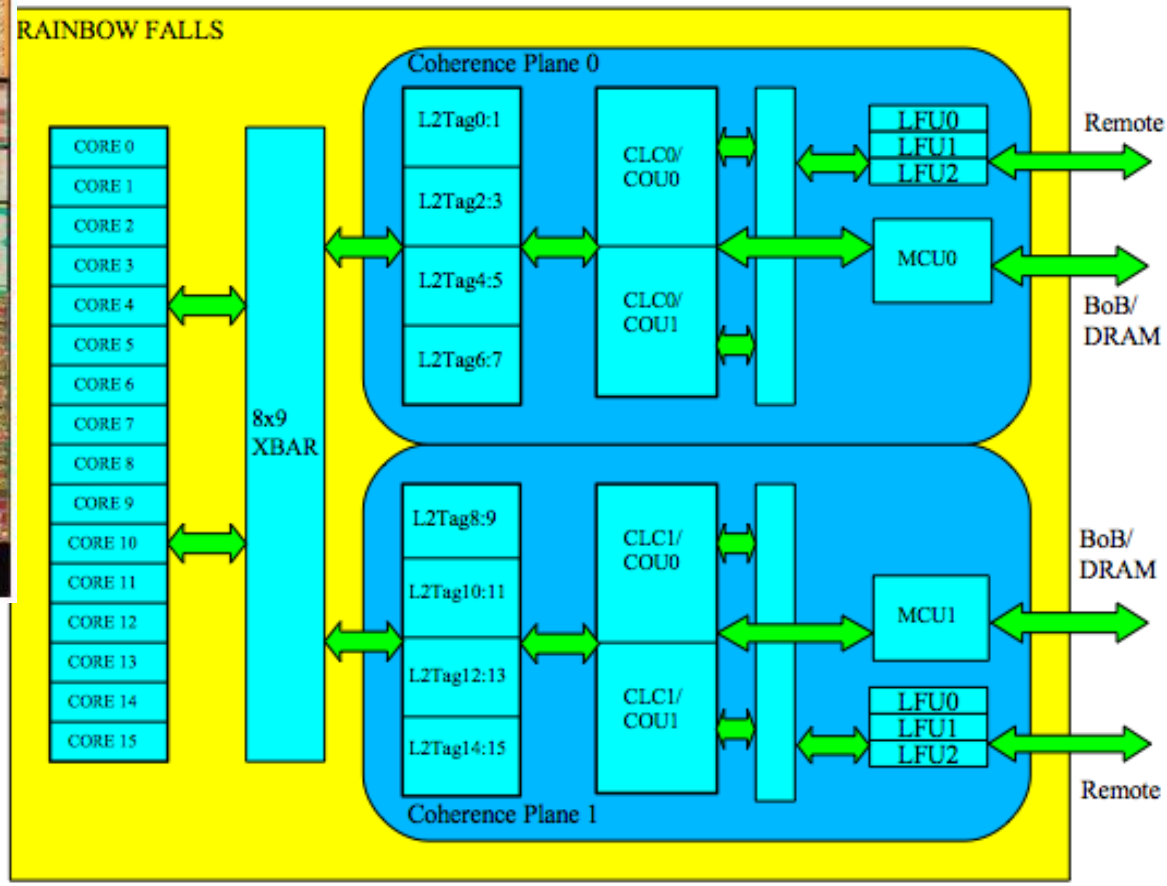
Oracle/Sun Niagara processors

- Target is datacenters running web servers and databases, with many concurrent requests
- Provide multiple simple cores each with multiple hardware threads, reduced energy/operation though much lower single thread performance

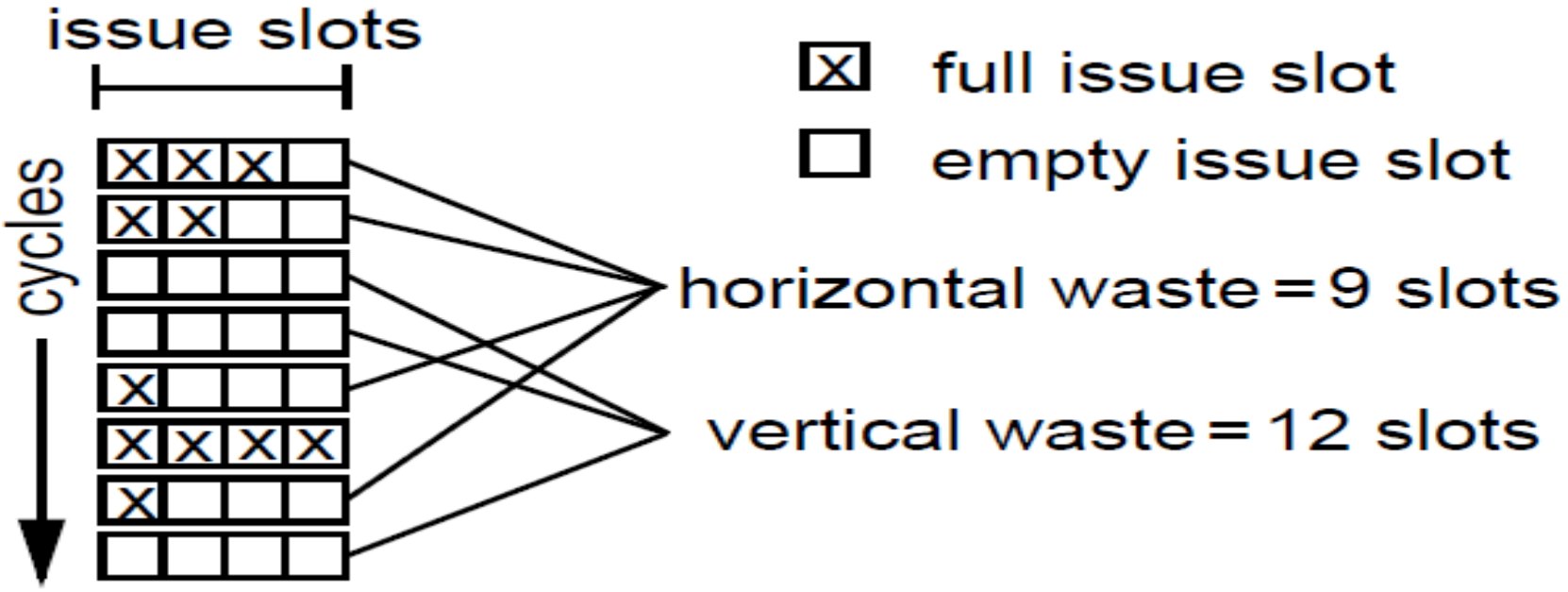
- Niagara-1 [2004], 8 cores, 4 threads/core
- Niagara-2 [2007], 8 cores, 8 threads/core
- Niagara-3 [2009], 16 cores, 8 threads/core
- T4 [2011], 8 cores, 8 threads/core
- T5 [2012], 16 cores, 8 threads/core

Oracle/Sun Niagara-3, “Rainbow Falls” 2009



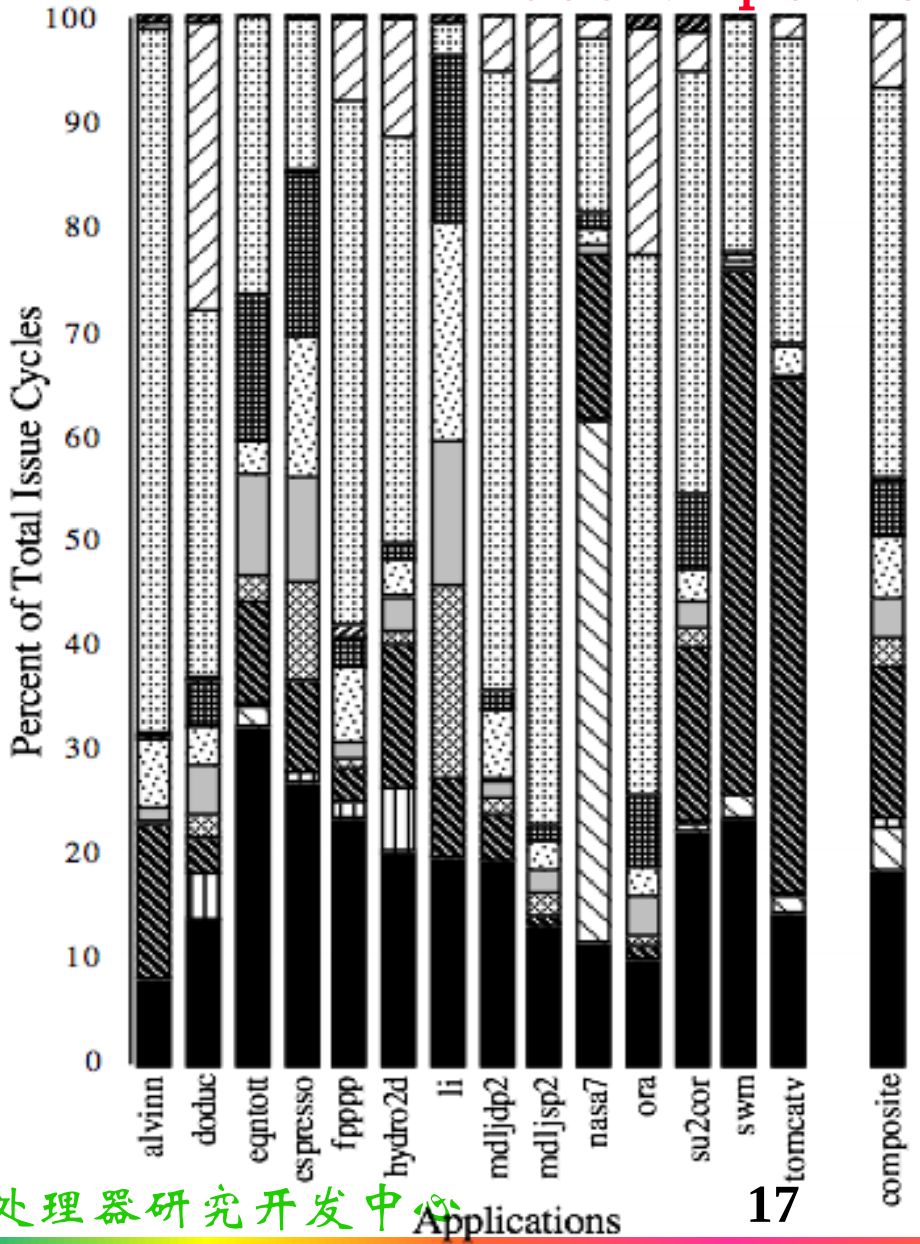


Vertical waste vs. Horizontal waste



- **Empty issue slots can be defined as either vertical waste or horizontal waste.**
- **Vertical waste is introduced when the processor issues no instructions in a cycle,**
- **horizontal waste when not all issue slots can be filled in a cycle.**

For most apps, most execution units lie idle in an OoO superscalar



For an 8-way superscalar.

From: Tullsen, Eggers, and Levy, "Simultaneous Multithreading: Maximizing On-chip Parallelism", ISCA 1995.

There is no dominant cause of wasted cycles (there appears to be no dominant solution)

1. Even an 8-issue superscalar architecture fails to sustain 1.5 instructions per cycle
2. Even if memory latencies are completely eliminated, we cannot achieve 40% utilization of this processor
   - A fine-grain multithreaded processor (capable of switching contexts every cycle at no cost) utilizes only about 40% of a wide superscalar, regardless of the number of threads.

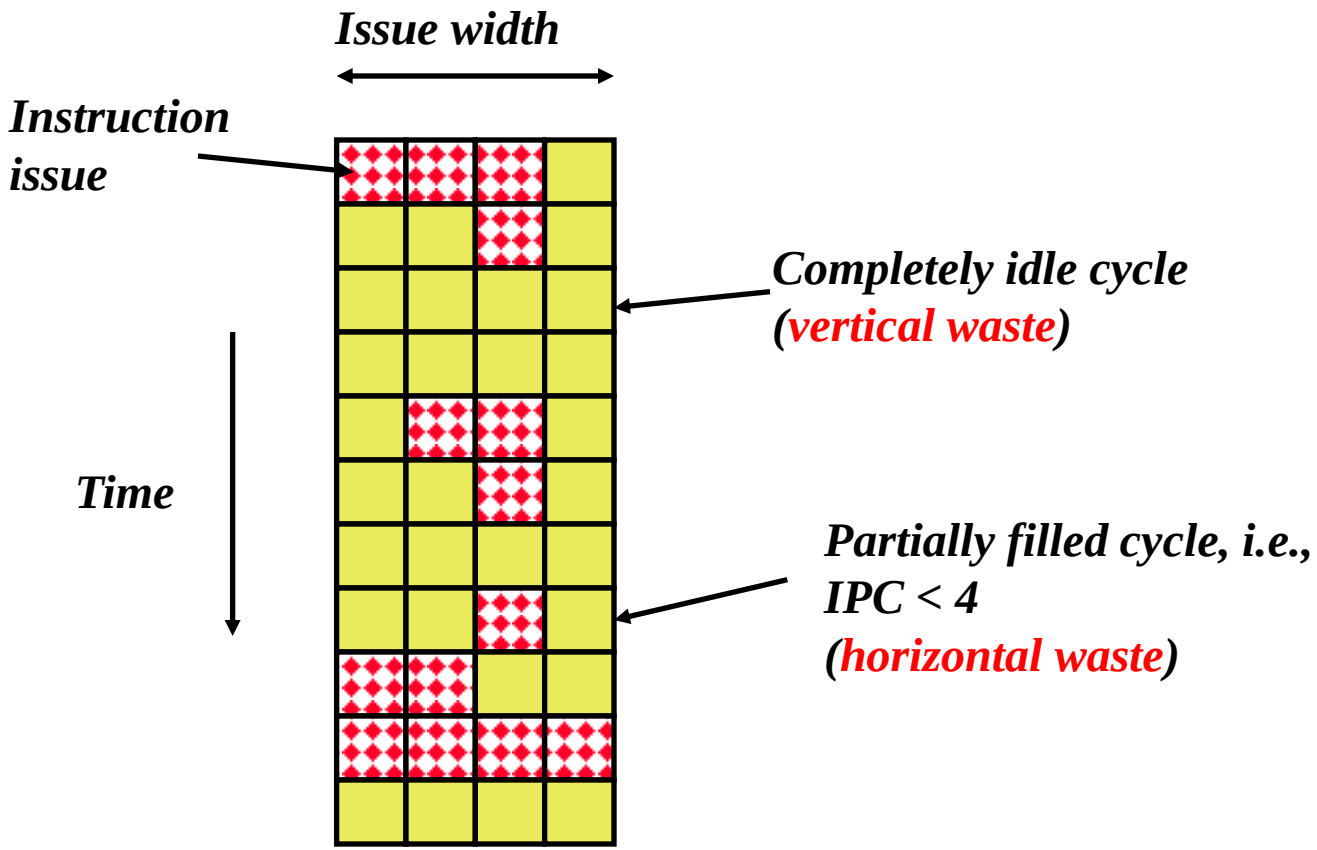
Source of Wasted Issue Slots and Possible Solutions

Source of Wasted Issue Slots	Possible Latency-Hiding or Latency-Reducing Technique
instruction tlb miss, data tlb miss	decrease the TLB miss rates (e.g., increase the TLB sizes); hardware instruction prefetching; hardware or software data prefetching; faster servicing of TLB misses
I cache miss	larger, more associative, or faster instruction cache hierarchy; hardware instruction prefetching
D cache miss	larger, more associative, or faster data cache hierarchy; hardware or software prefetching; improved instruction scheduling; more sophisticated dynamic execution
branch misprediction	improved branch prediction scheme; lower branch misprediction penalty
control hazard	speculative execution; more aggressive if-conversion
load delays (first-level cache hits)	shorter load latency; improved instruction scheduling; dynamic scheduling
short integer delay	improved instruction scheduling
long integer, short fp, long fp delays	(multiply is the only long integer operation, divide is the only long floating point operation) shorter latencies; improved instruction scheduling
memory conflict	(accesses to the same memory location in a single cycle) improved instruction scheduling

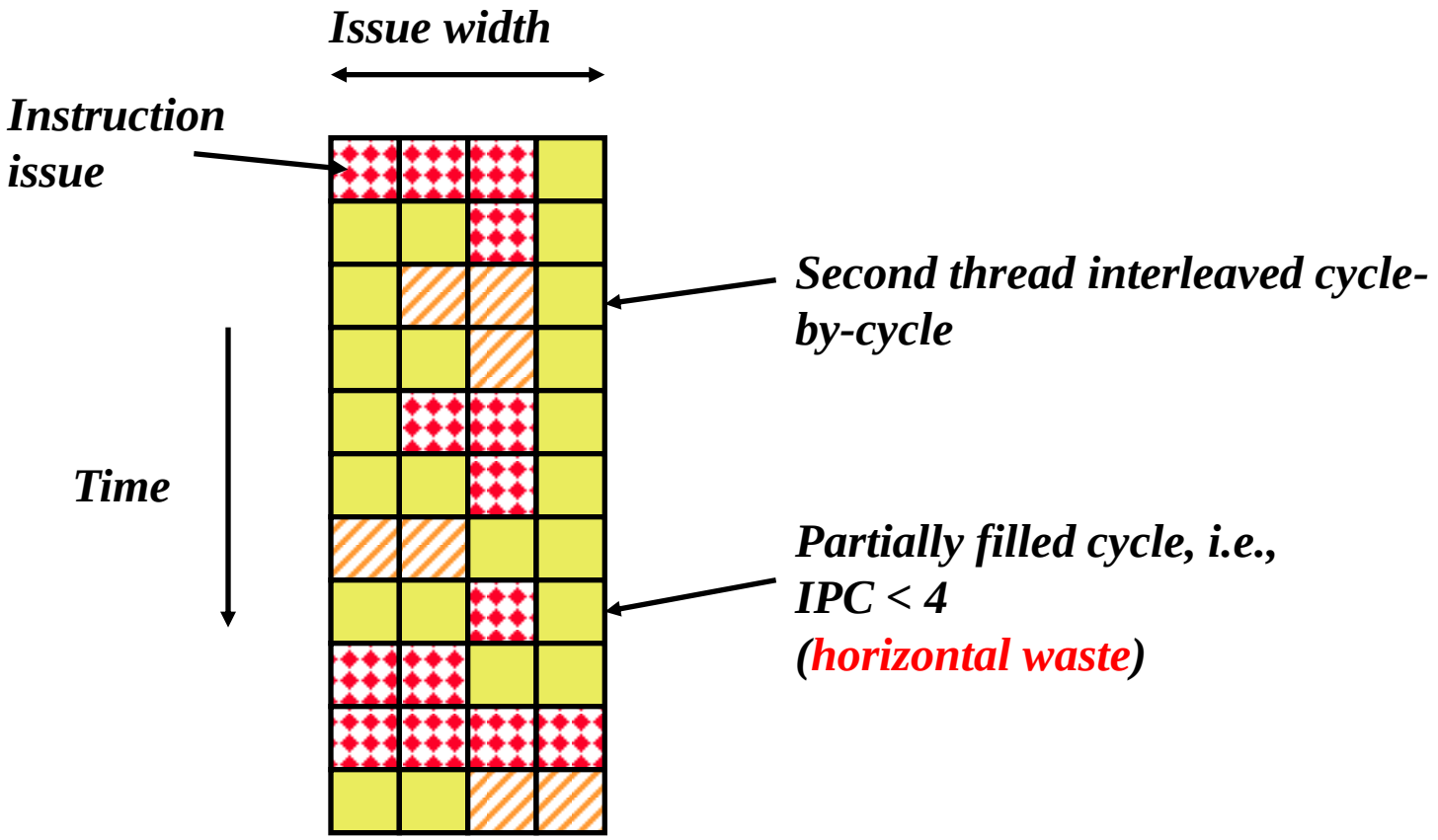
Simultaneous Multithreading (SMT) for OoO Superscalars

- The objective of SMT is to substantially increase processor utilization in the face of both long memory latencies and limited available parallelism per thread.
- SMT uses fine-grain control already present inside an OoO superscalar to allow instructions from multiple threads to enter execution on same clock cycle. Gives better utilization of machine resources.

Superscalar Machine Efficiency

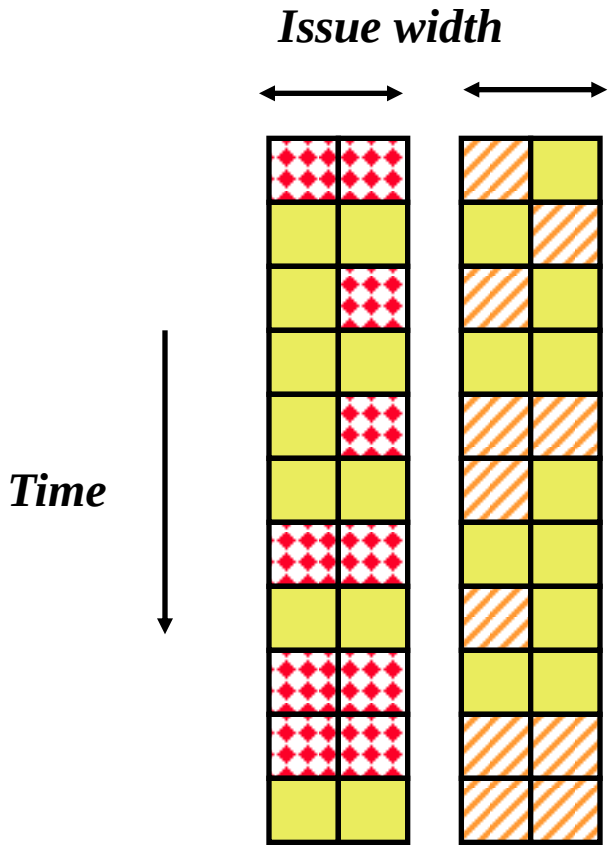


Vertical Multithreading



- What is the effect of cycle-by-cycle interleaving?
  - removes vertical waste, but leaves some horizontal waste

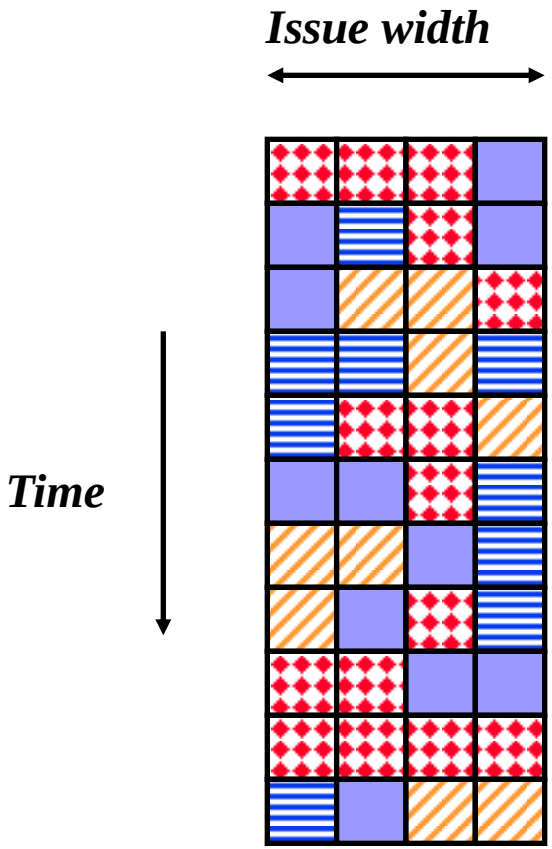
Chip Multiprocessing (CMP)



- What is the effect of splitting into multiple processors?
  - reduces horizontal waste,
  - leaves some vertical waste, and
  - puts upper limit on peak throughput of each thread.

Ideal Superscalar Multithreading

[Tullsen, Eggers, Levy, UW, 1995]



- Interleave multiple threads to multiple issue slots with no restrictions

Key hardware complexity features of the various models (H=high complexity).

Model	Register Ports	Inter-inst Dependence Checking	Forwarding Logic	Instruction Scheduling onto FUs	Notes
Fine-Grain	H	H	H/L*	L	Scheduling independent of other threads.
SM:Single Issue	L	None	H	H	
SM:Dual Issue	M	L	H	H	
SM:Four Issue	M	M	H	H	
SM:Limited Connection	M	M	M	M	No forwarding between FUs of same type; scheduling is independent of other FUs
SM:Full Simultaneous Issue	H	H	H	H	Most complex, highest performance

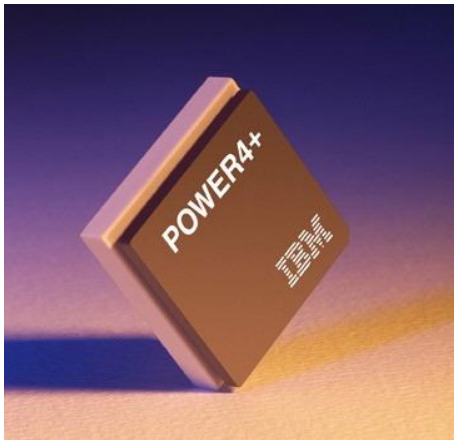
* We have modeled this scheme with all forwarding intact, but forwarding could be eliminated, requiring more threads for maximum performance

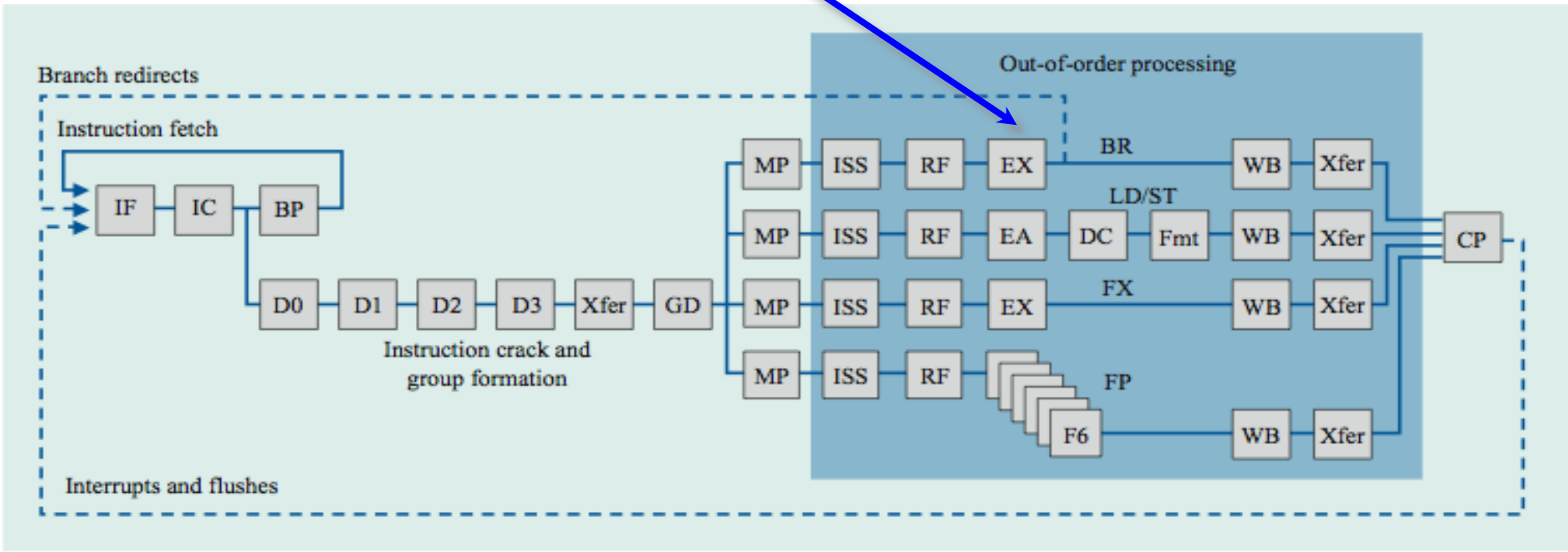
O-o-O Simultaneous Multithreading
[Tullsen, Eggers, Emer, Levy, Stamm, Lo, DEC/UW, 1996]

- Add multiple contexts and fetch engines and allow instructions fetched from different threads to issue simultaneously
- Utilize wide out-of-order superscalar processor issue queue to find instructions to issue from multiple threads
- OOO instruction window already has most of the circuitry required to schedule from multiple threads
- Any single thread can utilize whole machine

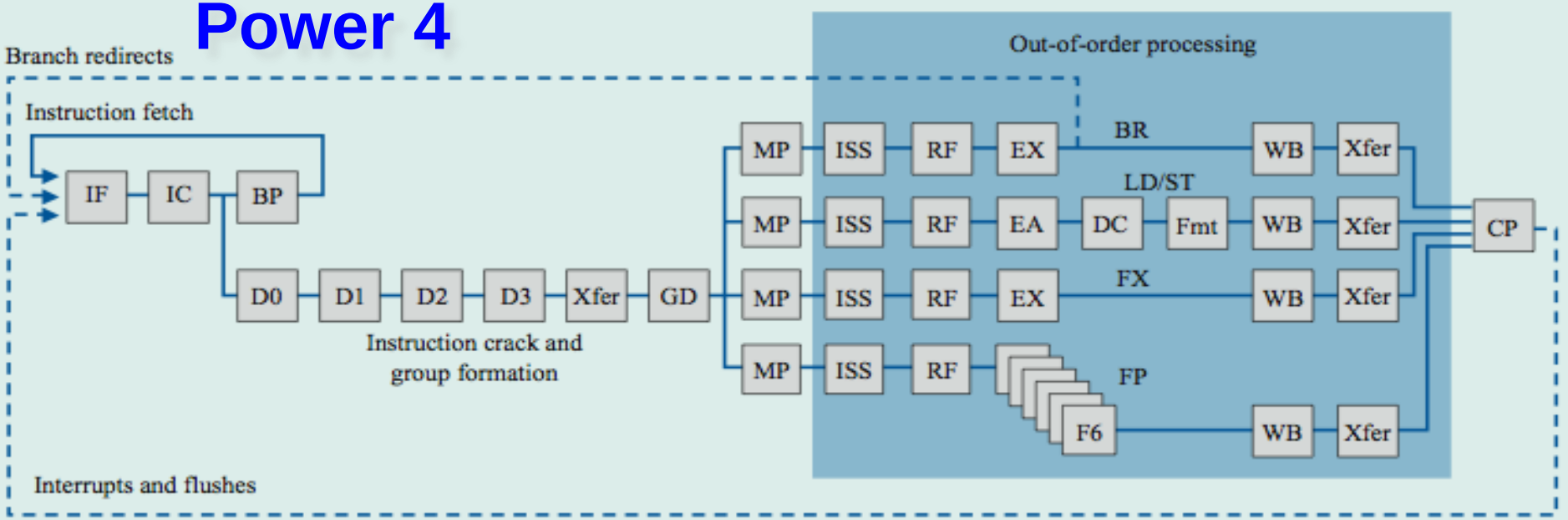
IBM Power 4

Single-threaded predecessor to Power 5. 8 execution units in out-of-order engine, each may issue an instruction each cycle.



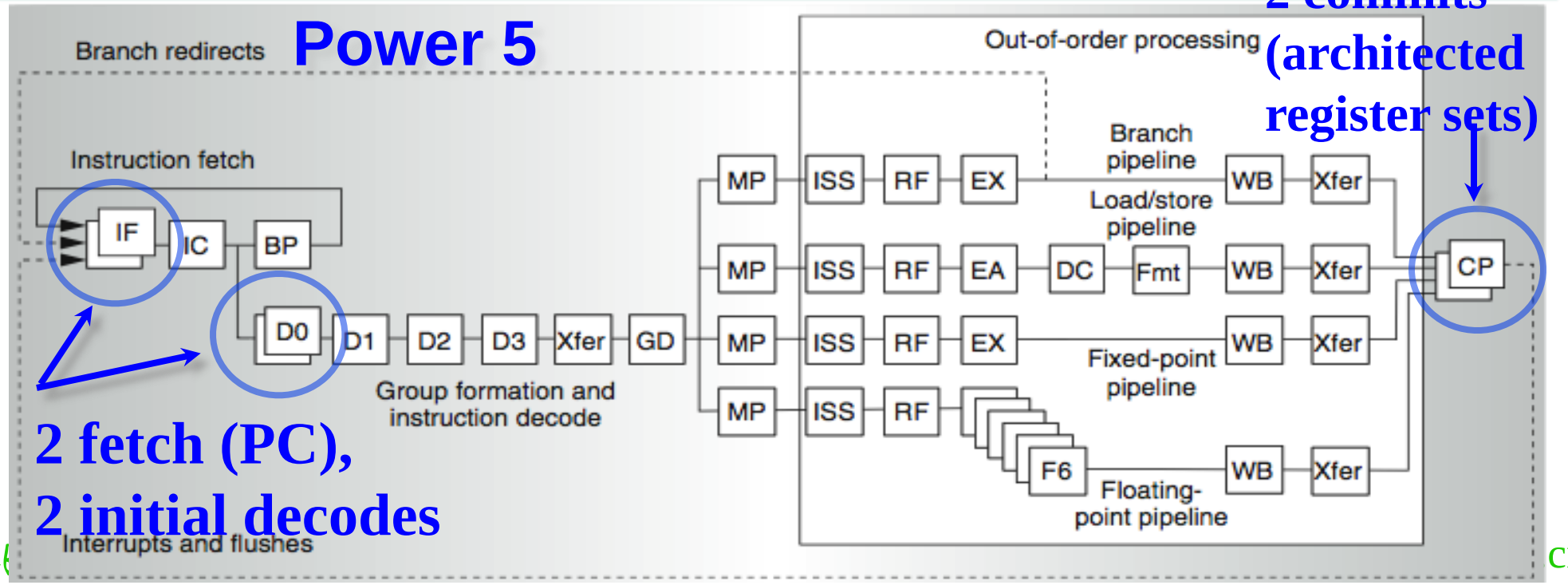


Power 4

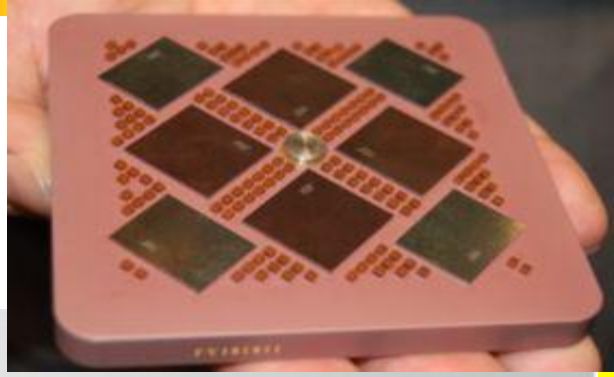


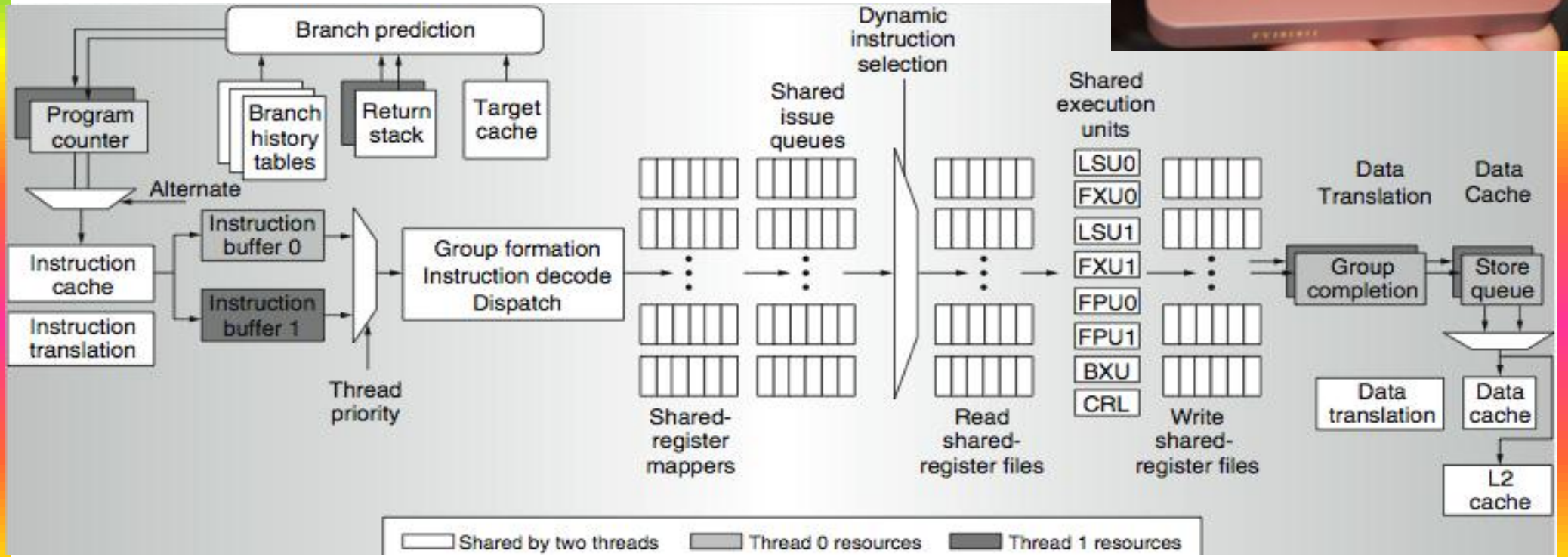
Power 5

2 commits (architected register sets)



Power 5 data flow ...





Why only 2 threads? With 4, one of the shared resources (physical registers, cache, memory bandwidth) would be prone to bottleneck

Changes in Power 5 to support SMT

- Increased associativity of L1 instruction cache and the instruction address translation buffers
- Added per-thread load and store queues
- Increased size of the L2 (1.92 vs. 1.44 MB) and L3 caches
- Added separate instruction prefetch and buffering per thread
- Increased the number of virtual registers from 152 to 240
- Increased the size of several issue queues
- The Power5 core is about 24% larger than the Power4 core because of the addition of SMT support

Pentium-4 Hyperthreading (2002)

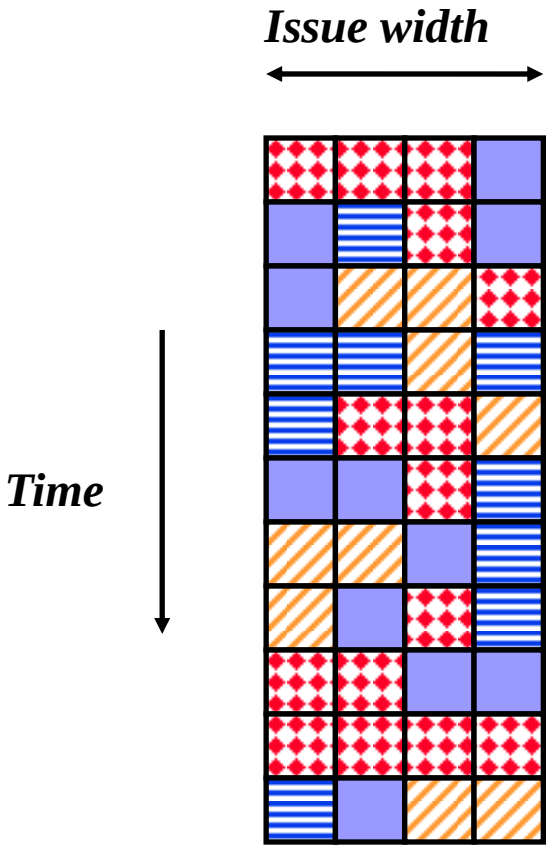
- First commercial SMT design (2-way SMT)
  - Hyperthreading == SMT
- Logical processors share nearly all resources of the physical processor
  - Caches, execution units, branch predictors
- Die area overhead of hyperthreading ~ 5%
- When one logical processor is stalled, the other can make progress
  - No logical processor can use all entries in queues when two threads are active
- Processor running only one active software thread runs at approximately same speed with or without hyperthreading
- Hyperthreading dropped on OoO P6 based followons to Pentium-4 (Pentium-M, Core Duo, Core 2 Duo), until revived with Nehalem generation machines in 2008.
- Intel Atom (in-order x86 core) has two-way vertical multithreading

Initial Performance of SMT

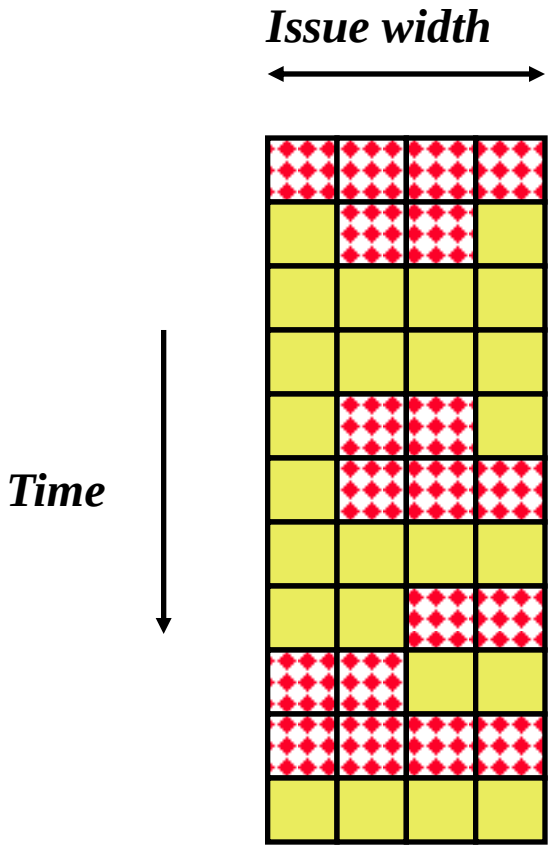
- Pentium 4 Extreme SMT yields 1.01 speedup for SPECint_rate benchmark and 1.07 for SPECfp_rate
  - Pentium 4 is dual threaded SMT
  - SPECRate requires that each SPEC benchmark be run against a vendor-selected number of copies of the same benchmark
- Running on Pentium 4 each of 26 SPEC benchmarks paired with every other (26^2 runs) speed-ups from 0.90 to 1.58; average was 1.20
- Power 5, 8-processor server 1.23 faster for SPECint_rate with SMT, 1.16 faster for SPECfp_rate
- Power 5 running 2 copies of each app speedup between 0.89 and 1.41
  - Most gained some
  - Fl.Pt. apps had most cache conflicts and least gains

SMT adaptation to parallelism type

For regions with high thread level parallelism (TLP) entire machine width is shared by all threads



For regions with low thread level parallelism (TLP) entire machine width is available for instruction level parallelism (ILP)



Summary: Multithreaded Categories