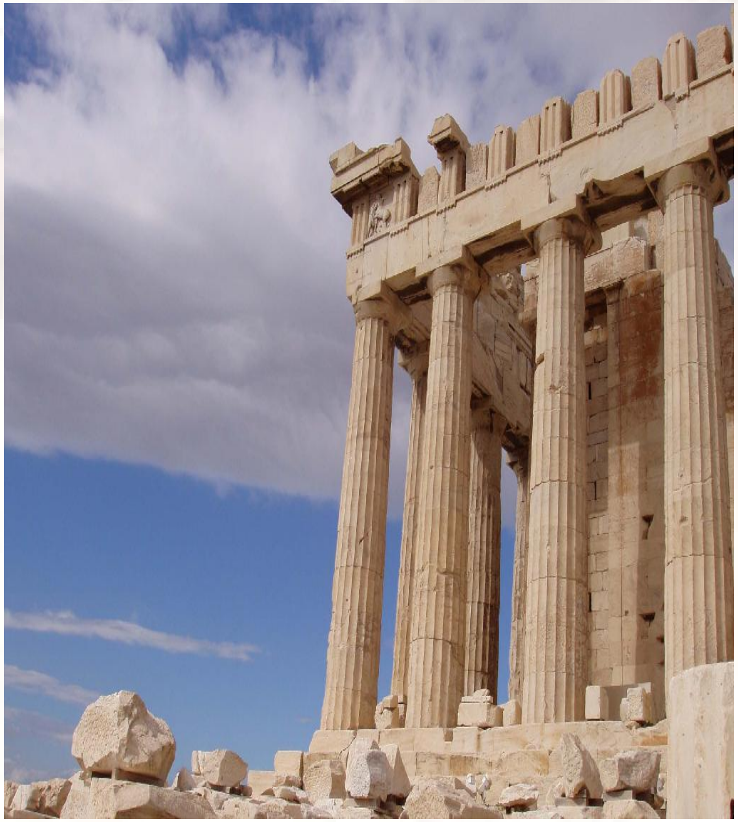
Ch3-6

MultiTreading

3.11 in edition 6



Pipeline Hazards

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

- Each instruction may depend on the next
  - Without bypassing, need interlocks

	t0	t1	t2	t3	t4	t5	t6	t7	t8	t9	t10	t11	t12	t13	t14
LW r1, 0(r2)	F	D	X	M	W										
LW r5, 12(r1)		F	D	D	D	D	X	M	W						
ADDI r5, r5, #12			F	F	F	F	D	D	D	D	X	M	W		
SW 12(r1), r5							F	F	F	F	D	D	D	D	

- Bypassing cannot completely eliminate interlocks or delay slots

Multithreaded Software

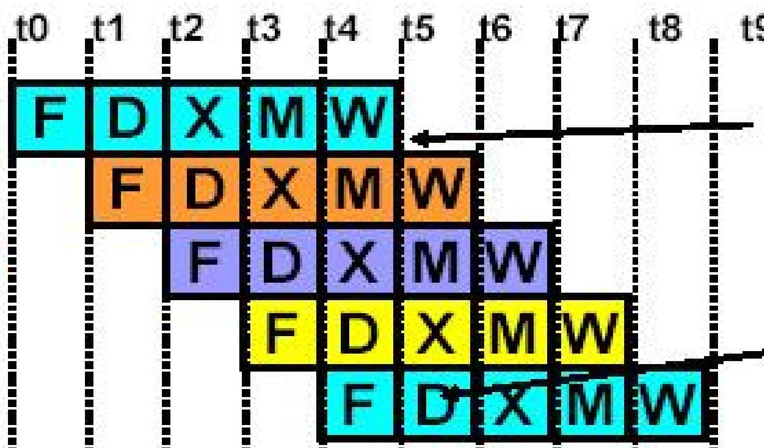
- Process
  - Each process has its unique address space
  - Can consist of several threads
- Thread – each thread has its unique execution context:
  - Its own PC + registers + stack
  - All threads within a process share same address space
  - Private heap is optional
- Multithreaded app's: process broken into threads
  - Increase concurrency
  - Partial blocking
  - Centralized (smaarter) resource management (by process)

Multithreading

- How can we guarantee no dependencies between instructions in a pipeline?
  - One way is to interleave execution of instructions from different program threads on same pipeline

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)



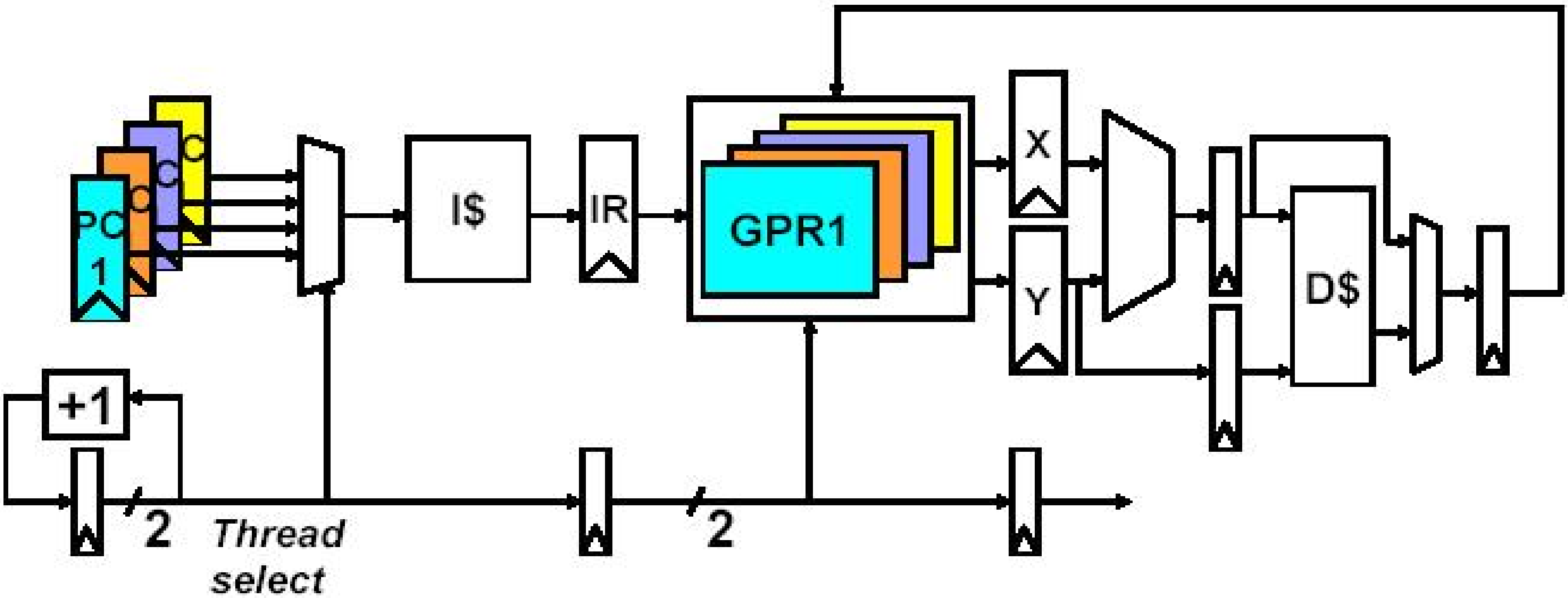
Last instruction in a thread always completes writeback before next instruction in same thread reads regfile

3

Multithreaded Architecture

- Processor capable of executing multiple software threads
  - Can execute “simultaneously”
    - Thread can be HW switched without OS control
  - Shared resource
    - Sharing -> better resource utilization -> better throughtput
  - Can belong to the same process – but not have to !

Simple Multithreaded Pipeline



- Have to carry thread select down pipeline to ensure correct state bits read/written at each pipe stage

Multithreading Costs

- Appears to software (including OS) as multiple slower CPUs

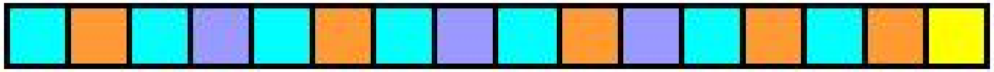
- Each thread requires its own user state
  - GPRs
  - PC

- Also, needs own OS control state
  - virtual memory page table base register
  - exception handling registers

- *Other costs?*

Thread scheduling policies

- Fixed interleave *(CDC 6600 PPUs, 1965)*
  - 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)

The Denelcor HEP was a uniform shared memory multiprocessor that used fine-grain multithreading to tolerate memory latency, synchronization latency, and even functional unit latency.

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 (1990-97)

- 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
  - CMOS version, MTA-2, 50W/processor

MTA Architecture

- Each processor supports 128 active hardware threads
  - 1 x 128 = 128 stream status word (SSW) registers,
  - 8 x 128 = 1024 branch-target registers,
  - 32 x 128 = 4096 general-purpose registers
- Three operations packed into 64-bit instruction (short VLIW)
  - One memory operation,
  - One arithmetic operation, plus
  - One arithmetic or branch operation
- Thread creation and termination instructions
- Explicit 3-bit "lookahead" field in instruction gives number of subsequent instructions (0-7) that are independent of this one
  - c.f. instruction grouping in VLIW
  - allows fewer threads to fill machine pipeline
  - used for variable-sized branch delay slots

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 when cache getting hits
- Just add a few threads to hide occasional cache miss latencies
- Swap threads on cache misses

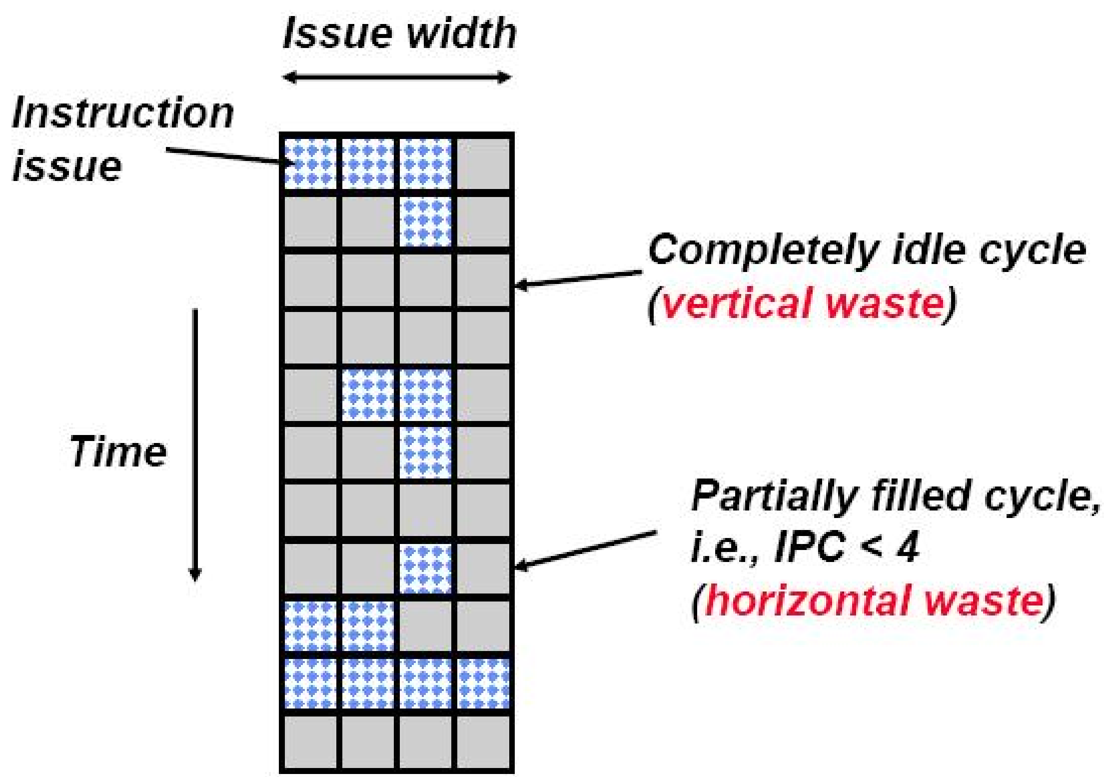
IBM Power RS64-IV (2000)

- 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

Multithreading design choices

- Fine-grained multithreading
  - Context switch among threads every cycle
- Coarse-grained multithreading
  - Context switch among threads every few cycles, e.g., on:
    - » Function unit data hazard,
    - » L1 miss,
    - » L2 miss...
- Why choose one style over another?
- Choice depends on
  - Context-switch overhead
  - Number of threads supported (due to per-thread state)
  - Expected application-level parallelism...

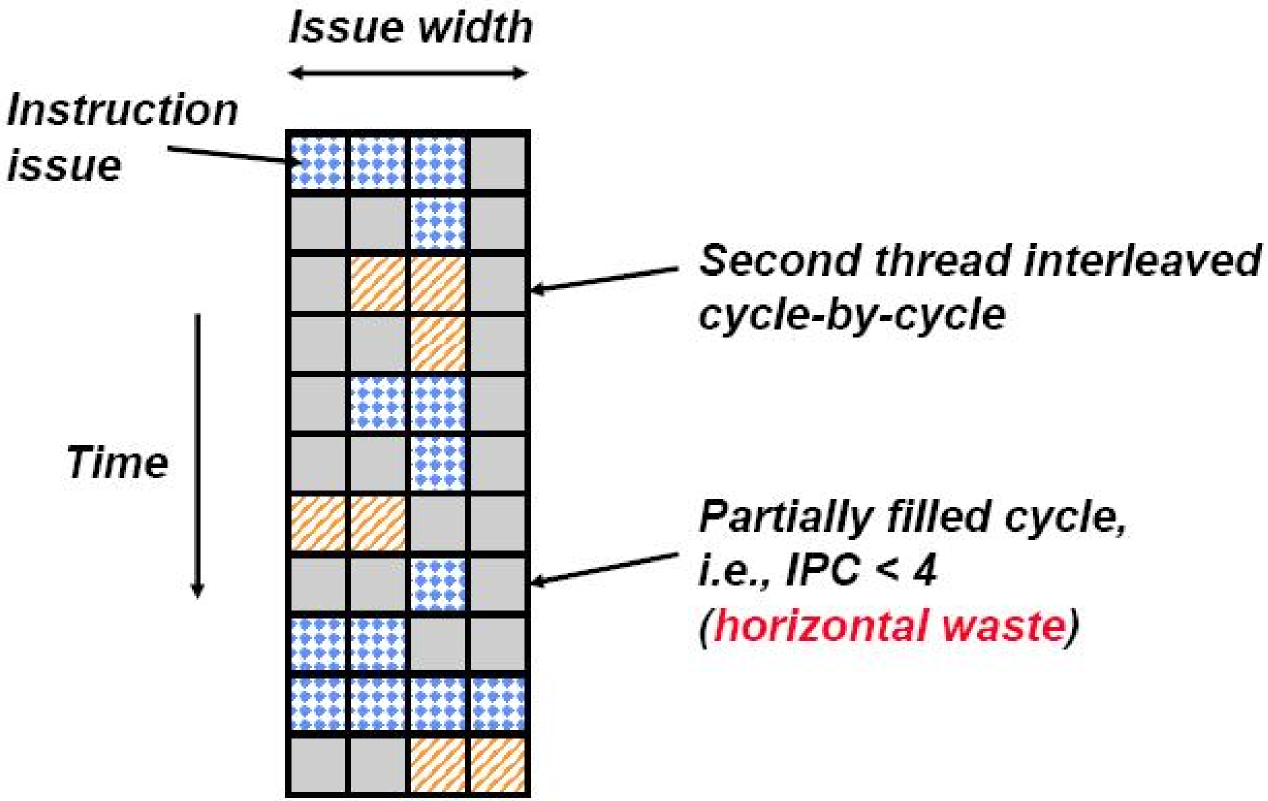
Superscalar Machine Efficiency



- *Why horizontal waste?*
- *Why vertical waste?*

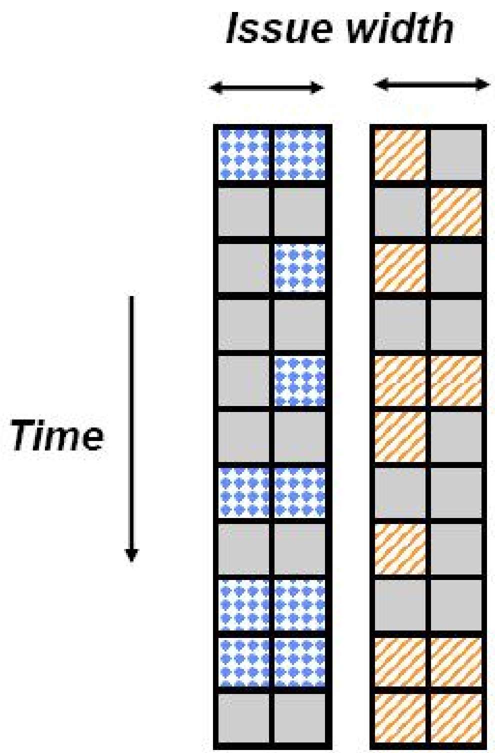
Cache misses

Vertical Multithreading



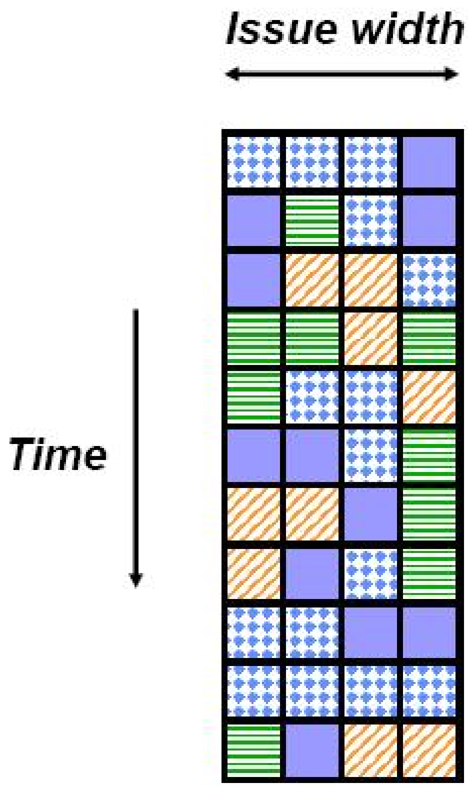
- What is the effect of cycle-by-cycle interleaving?
  - removes vertical waste, but leaves some horizontal waste

Chip Multiprocessing



- What is the effect of splitting into multiple processors?
  - eliminates horizontal waste,
  - leaves some vertical waste, and
  - caps peak throughput of each thread.

Ideal Superscalar Multithreading
[Tullsen, Eggers, Levy, UW, 1995]



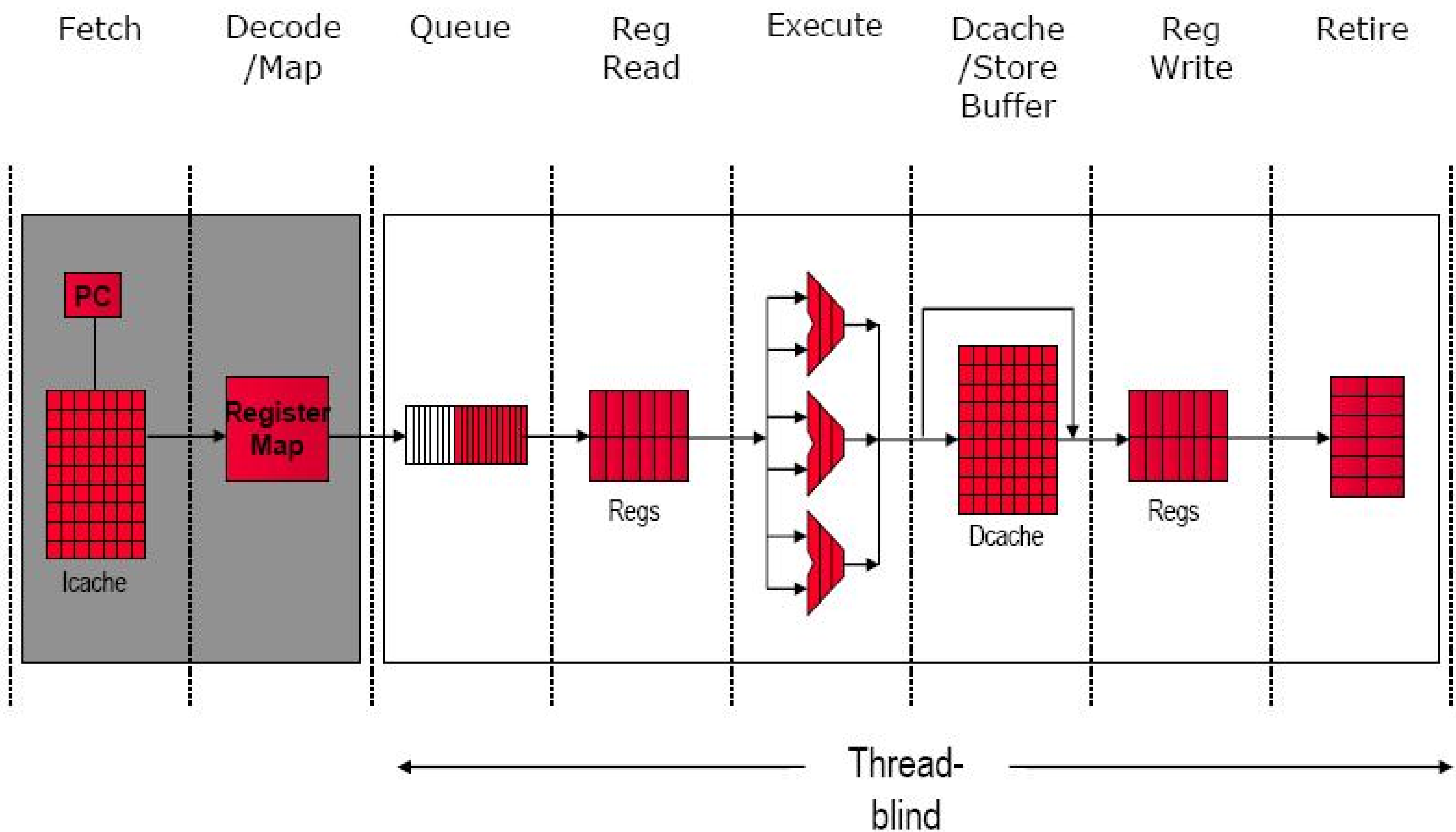
- Interleave multiple threads to multiple issue slots with no restrictions

O-o-O Simultaneous Multithreading

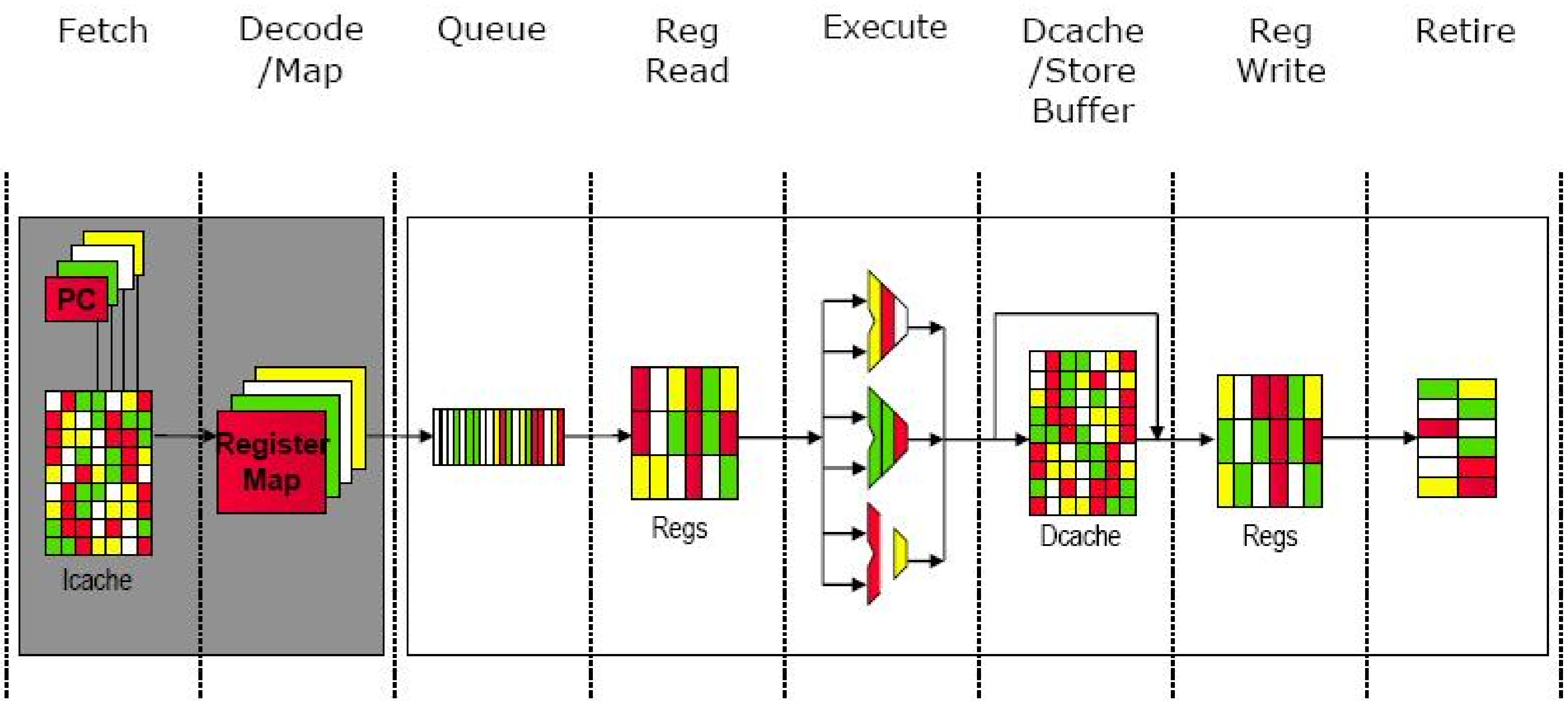
[Tullsen, Eggers, Emer, Levy, Stamm, Lo, DEC/UW, 1996]

- 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

Basic Out-of-order Pipeline



SMT Pipeline



Design Challenges in SMT

- Trade-off between fine-grained implementation performance and single-thread performance.
  - preferred thread : might sacrifices throughput
  - Less likely to have a mix of instructions from several threads.
  - Maximize single-thread performance, should fetch as far ahead as possible, and have the fetch unit free when a branch is mispredicted or miss occur in prefetch buffer.

Design Challenges in SMT

- A larger register file to hold multiple context
- Not affecting the clock cycle, such as in instruction issue, in instruction completion.
- Ensuring that Cache and TLB conflicts do not cause performance degradation.
-

Power 5 with SMT support vs Power 4

- Increasing the associativity of L1 icache and iTLB
- Adding per-thread load and store queues
- Increasing the size of L2 and L3
- Adding separate instruction prefetch and buffering
- Increasing virtual registers from 152 to 240
- Increasing size of several issue queues

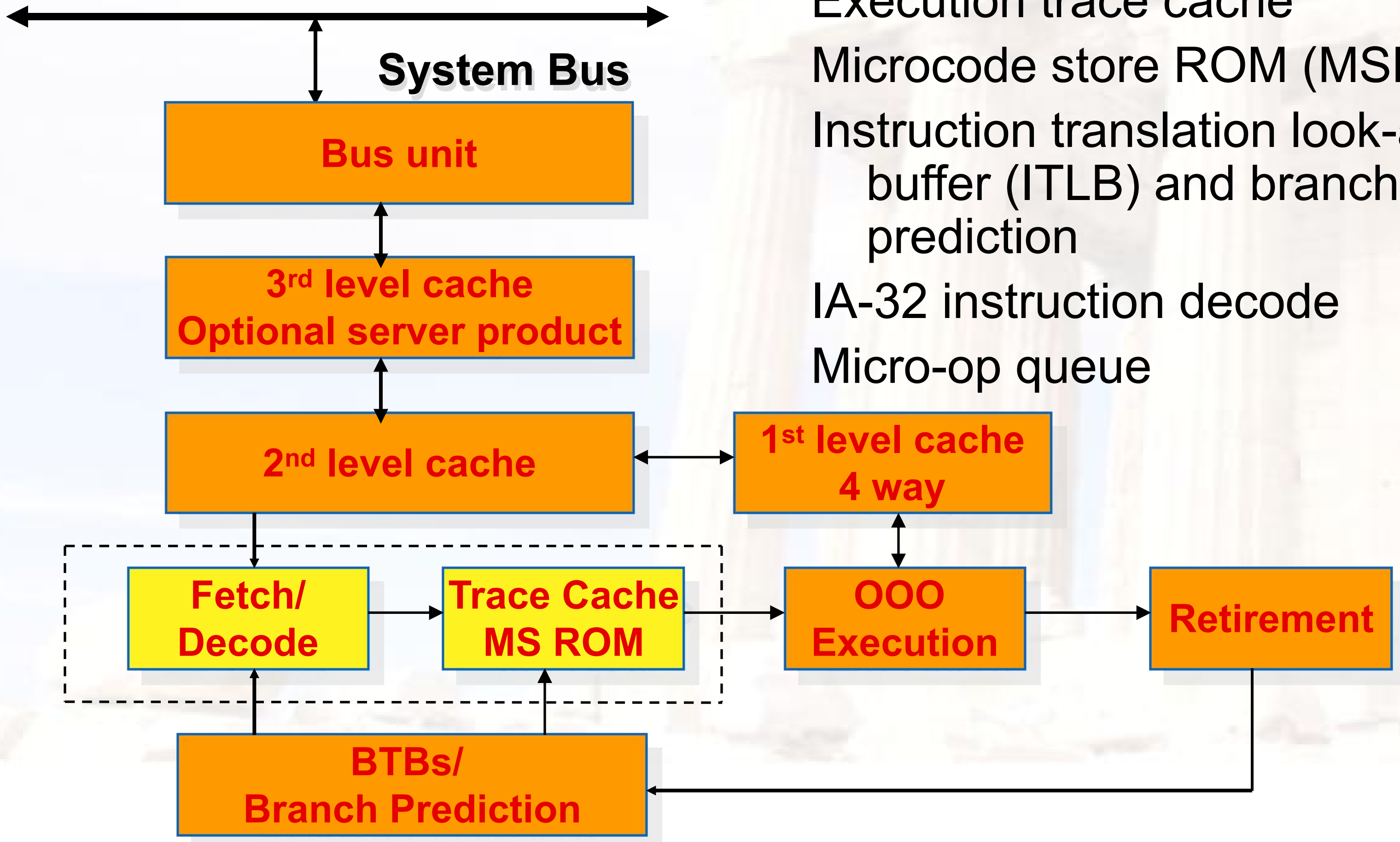
Pentium-4 Hyperthreading

- 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

HT Technology Architecture

Front End

- Execution trace cache
- Microcode store ROM (MSROM)
- Instruction translation look-aside buffer (ITLB) and branch prediction
- IA-32 instruction decode
- Micro-op queue



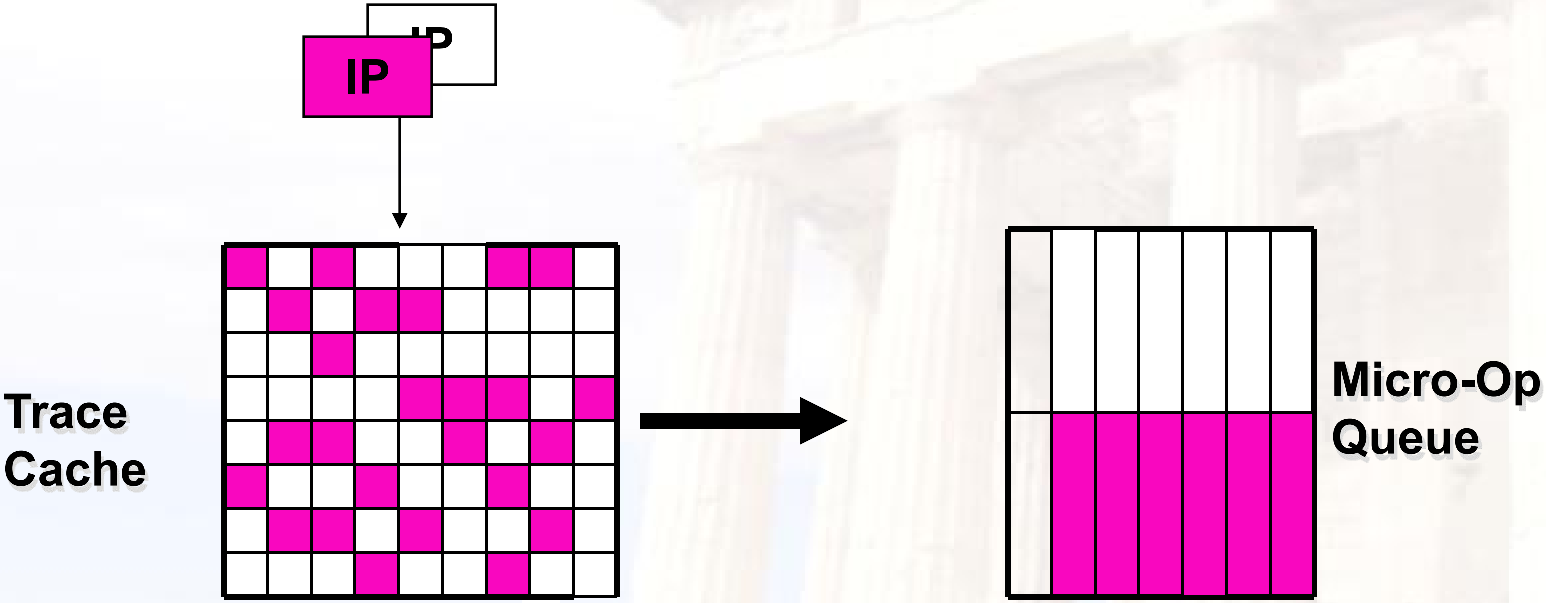
ZHEJIANG UNIVERSITY 浙江大學

Front End

- Responsible for delivering instruction to the later pipe stages
- Trace cache hit
  - When the requested instruction trace is present in trace cache
- Trace cache miss
  - Requested instruction is brought in the trace cache from L2 cache

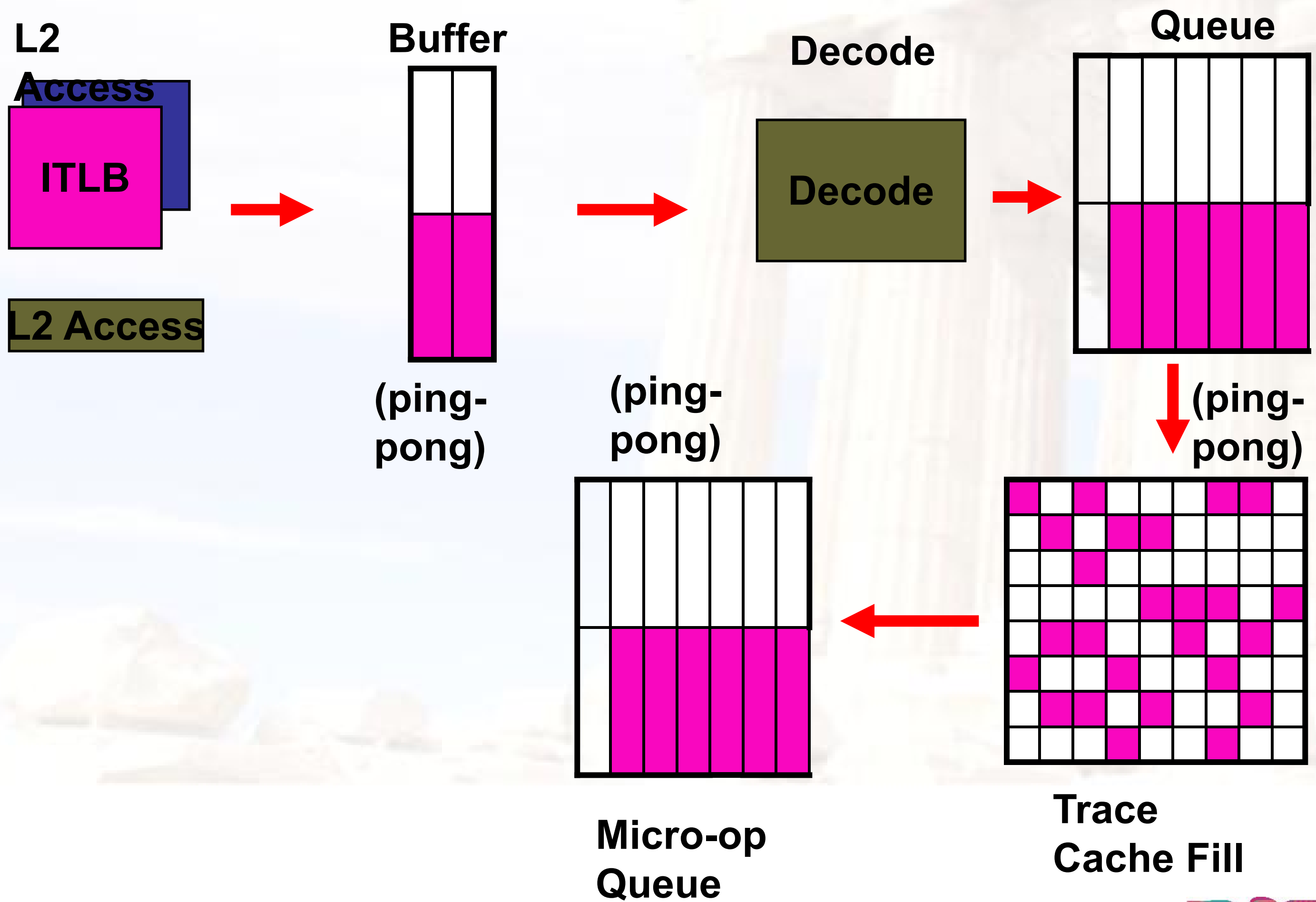
Trace Cache (TC) Hit

Front End



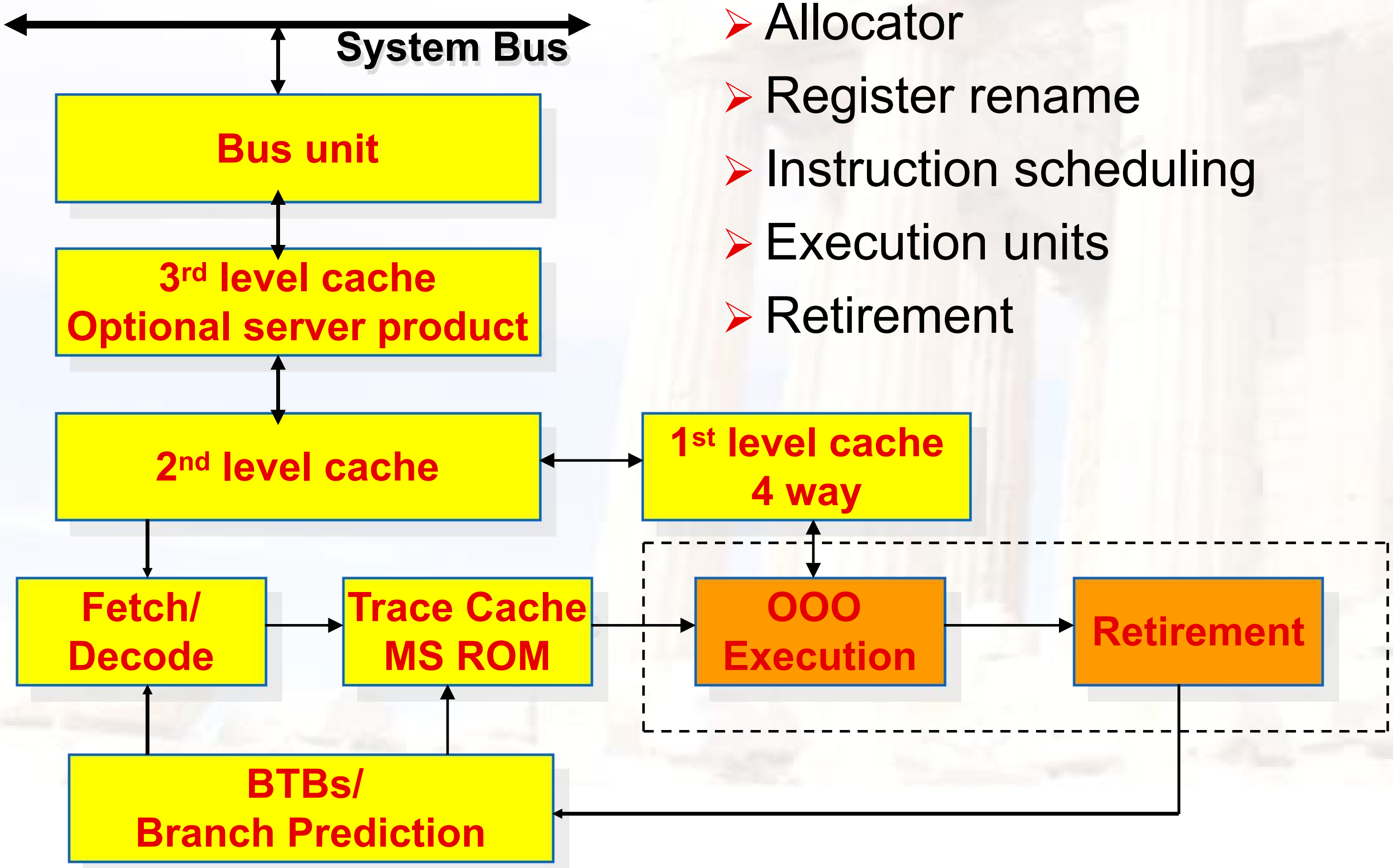
- Two sets of instruction pointers
- Two logical processors arbitrates access to TC every clock cycle
- If one logical processor stalled, then other logical processor can use the full bandwidth of TC

Trace Cache Miss
Front End



ZHEJIANG UNIVERSITY

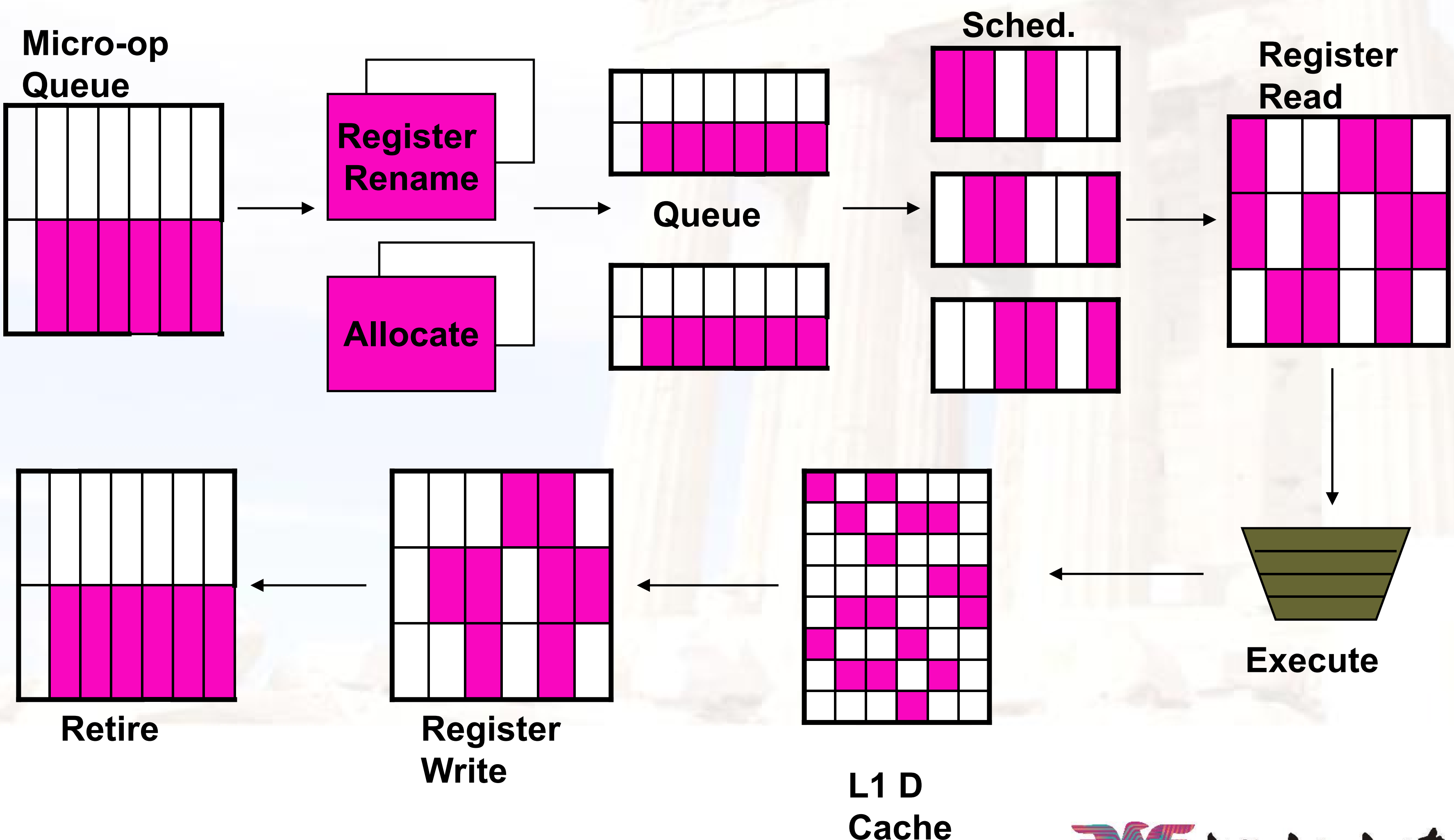
HT Technology Architecture



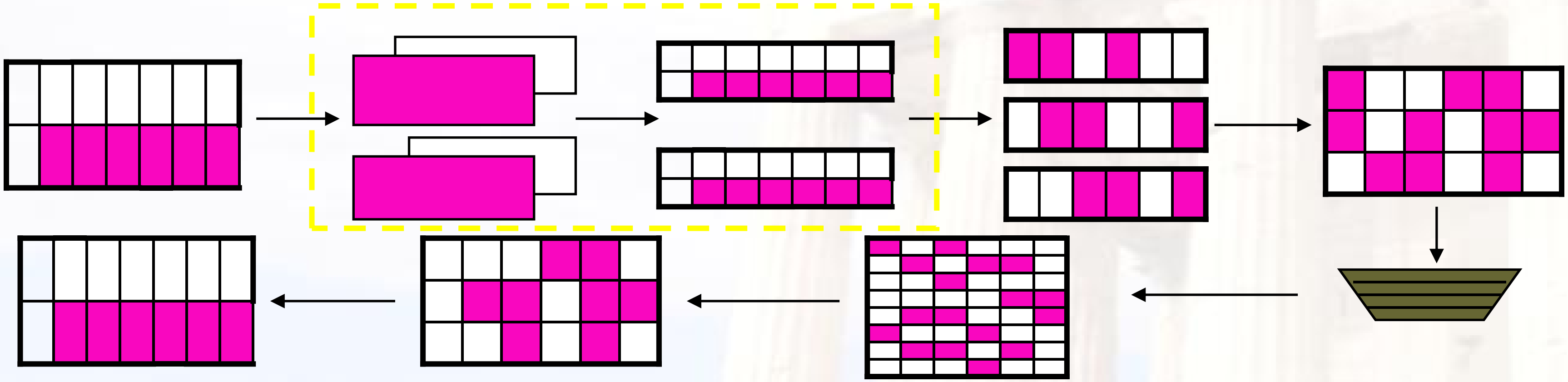
Back End

- Allocator
- Register rename
- Instruction scheduling
- Execution units
- Retirement

Detailed Pipeline



Allocator
Back End



Allocates many of the key machine buffers including

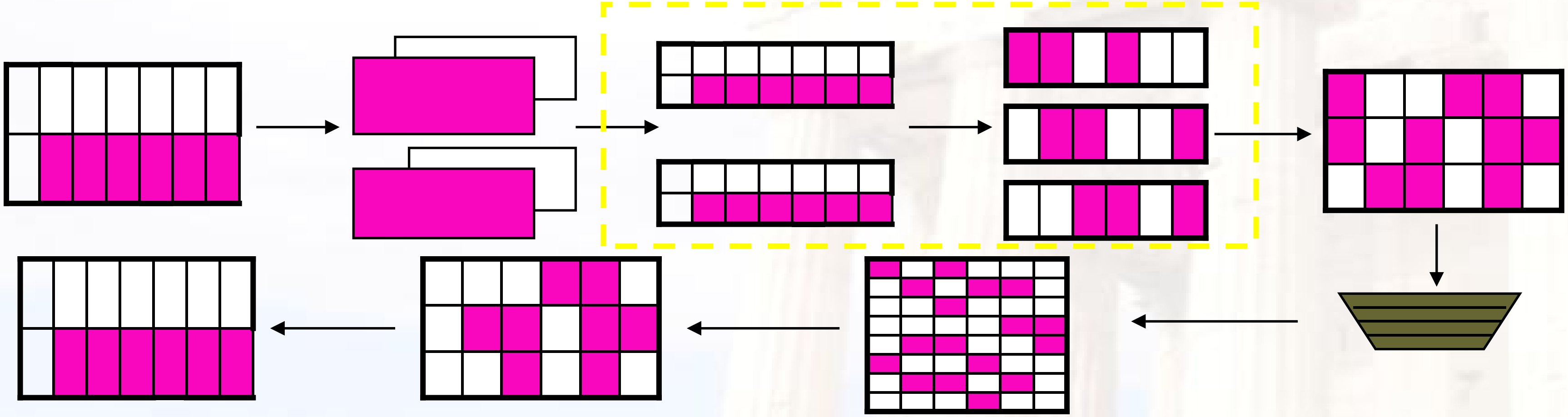
- 126 re-order buffer entries
- 128 integer and floating point physical registers
- 48 load 24 store buffers
- When HT Technology is enabled, each logical processor can use maximum of 63 re-order buffer entries, 24 load buffers and 12 store buffers

ZHEJIANG UNIVERSITY

Register Rename
Back End

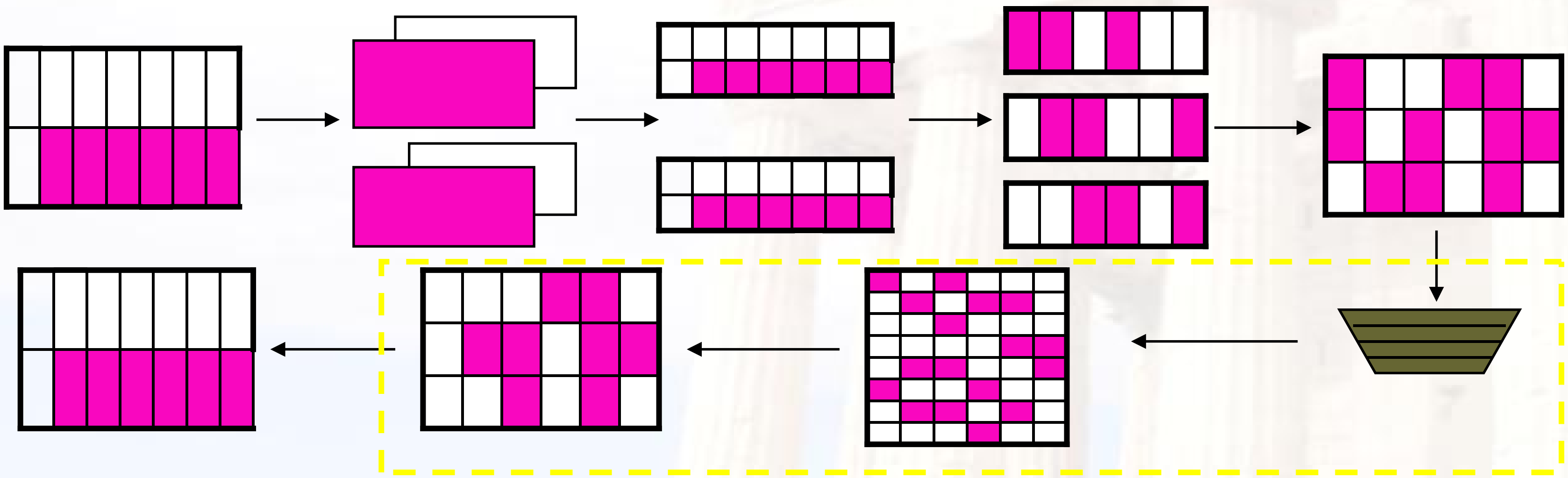
- There are two Register Alias Tables (RAT) for each processor
- Rename done in parallel with allocator logic
- Once done, micro-ops are placed into two sets of queues: MIAQ(memory instruction address queue) and GIAQ(general instruction address queue)
- Queues are partitioned such that micro-ops from each logical processor can use at most half the entries

Instruction Scheduling
Back End



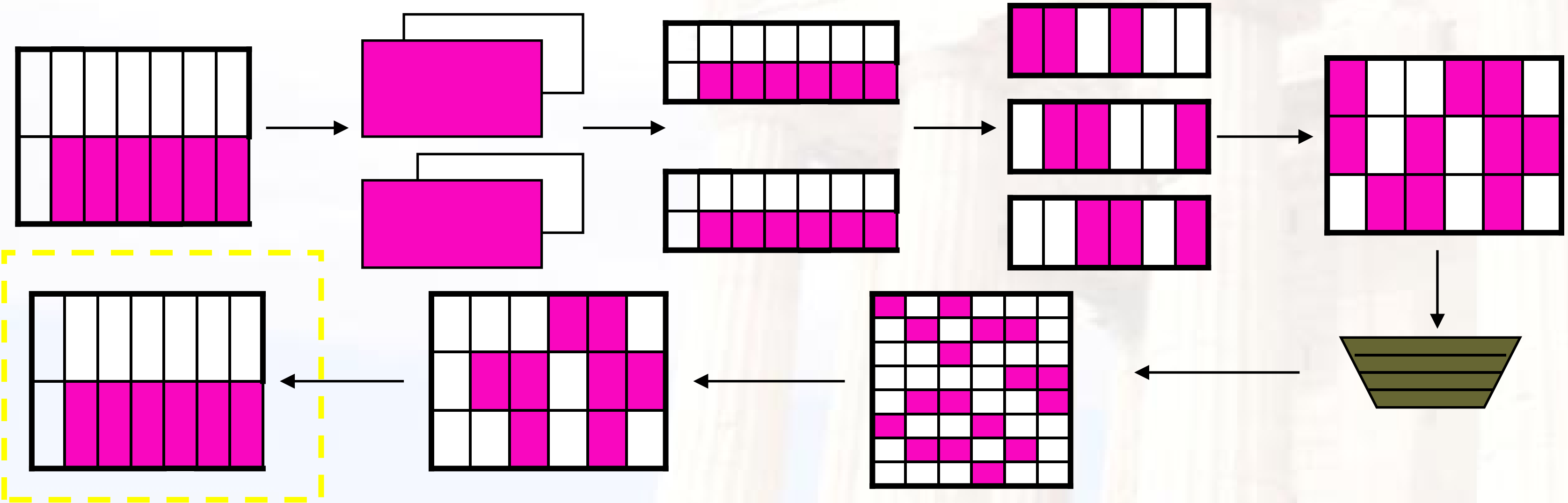
- Five schedulers are used to schedule different types of micro-ops
- Both MIAQ and GIAQ send micro-op to five schedulers as fast as they can
- Each scheduler has its own scheduler queue
- Micro-ops are simply evaluated based on dependent inputs and availability of execution resources
- To avoid deadlock, there is a limit on number of active entries that a logical processor can have in scheduler's queue

Execution units
Back End



- After execution, micro-ops are placed in re-order buffer. It decouples execution stage from retirement stage
- Re-order buffer (ROB) is partitioned such that each logical processor can use half entries

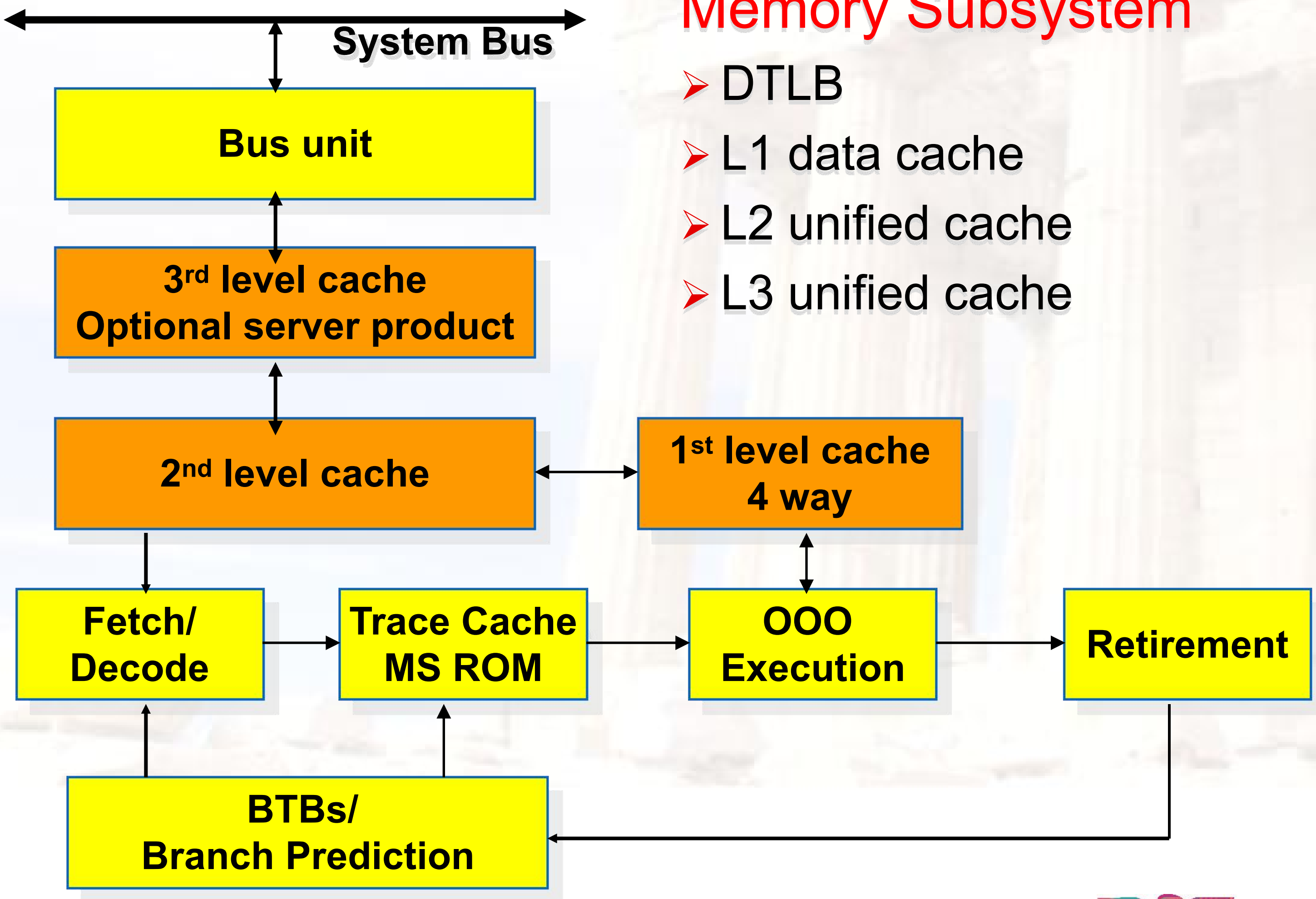
Retirement
Back End



- Tracks when micro-ops from two logical processors are ready to be retired, then retires the micro-ops by alternating between two logical processors

ZHEJIANG UNIVERSITY

HT Technology Architecture



Memory Subsystem

- DTLB
- L1 data cache
- L2 unified cache
- L3 unified cache

DTLB
Memory Subsystem

- Tagged resource
- Each entry in dual translation lookaside buffer (DTLB) includes a logical processor ID tag
- Each processor has reservation register to ensure fairness and forward progress in processing DTLB misses

Memory Subsystem
L1 Data Cache, L2 Cache, L3 Cache

- Both logical processors can share all entries in all three levels of cache

Bus

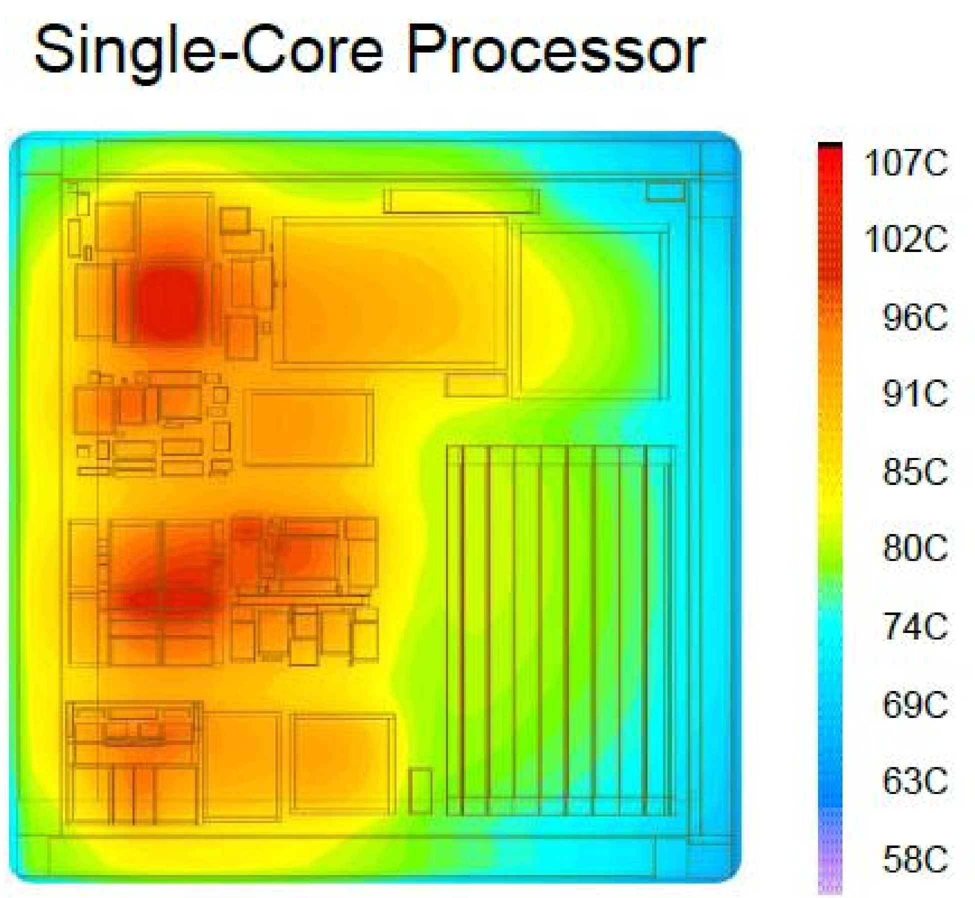
- Logical processors are treated on first-come first-serve basis. Priority is not given to any logical processor
- Distinction between requests from two logical processors are maintained
- Requests to APIC and interrupt delivery resources are unique and separate per logical processor

Sun T1 Multiprocessor

- Focus on Thread Level Parallelism
- Commercial application oriented
- Small single-issue core employ multithreading
- Multi core on chip

Single Thread Design

- L2 cache is getting larger with less performance contribution
- Power efficiency decrease
- Hot-spot
- Lower processor utilization

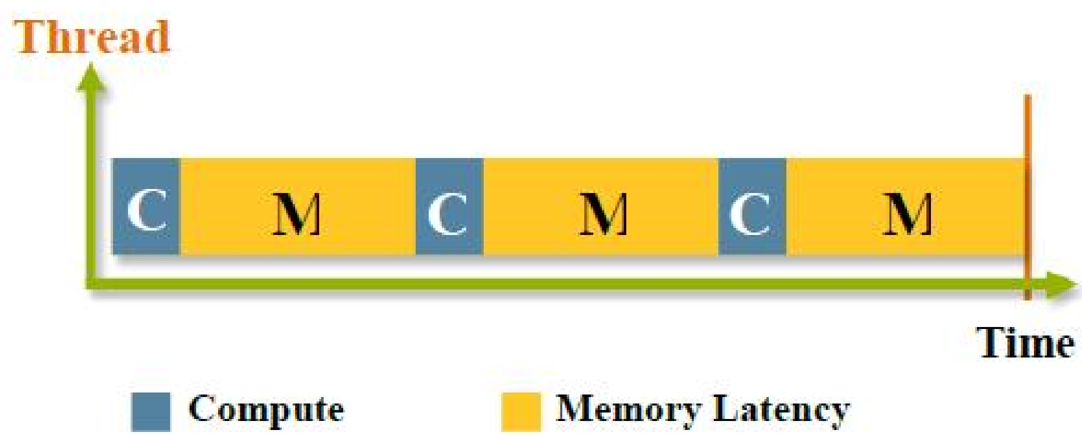


Single Thread design multiple issue

Up to 85% Cycles Spent Waiting for Memory

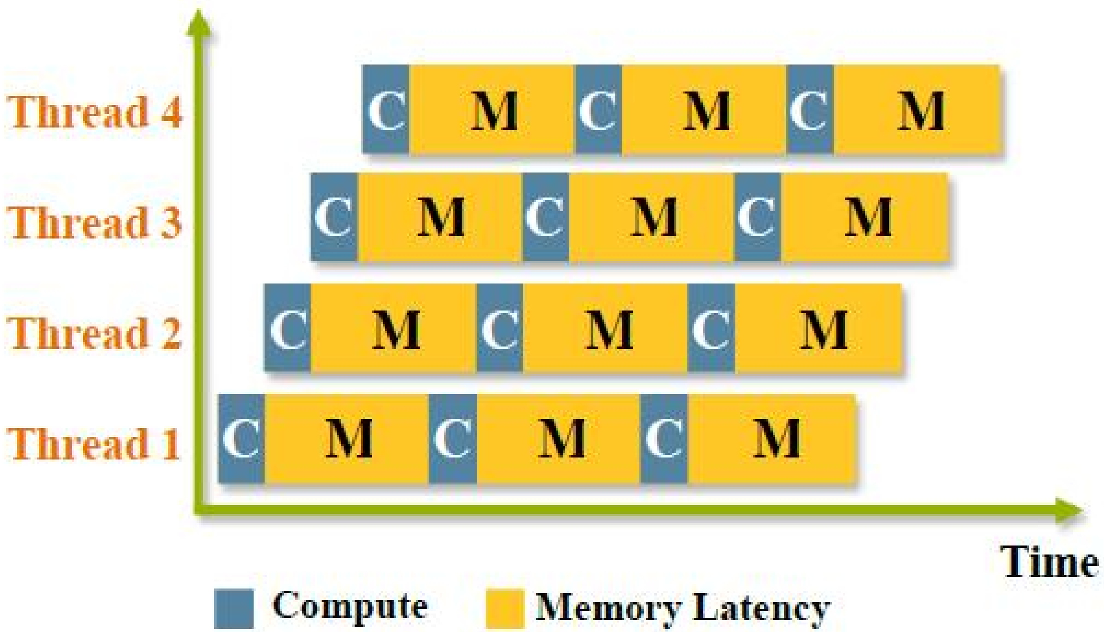
Single Threaded Performance

Typical Utilization of Processor: 15–25%

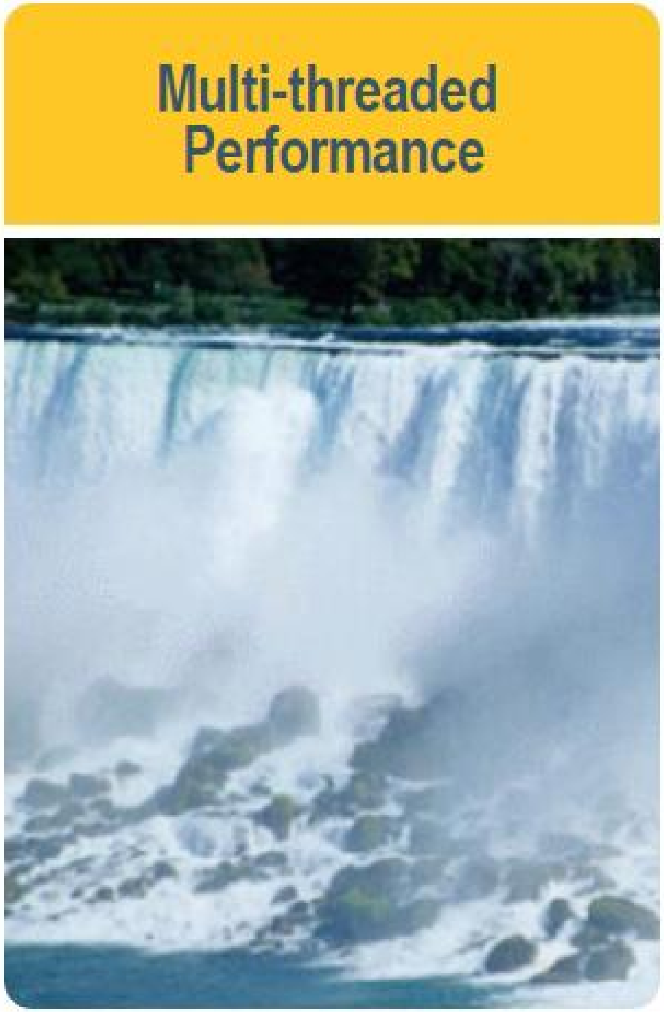


Hardware Multi Threading

Utilization: Up to 85%*

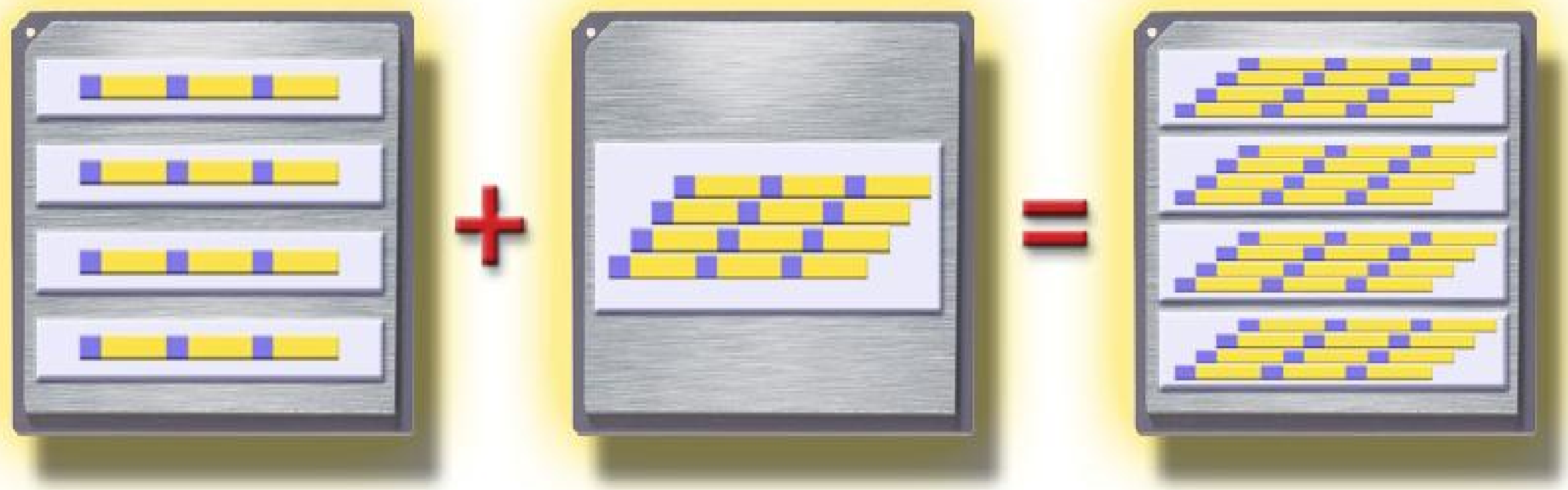


* based on example of UltraSPARC T1



ZHEJIANG UNIVERSITY

Chip MultiThreading

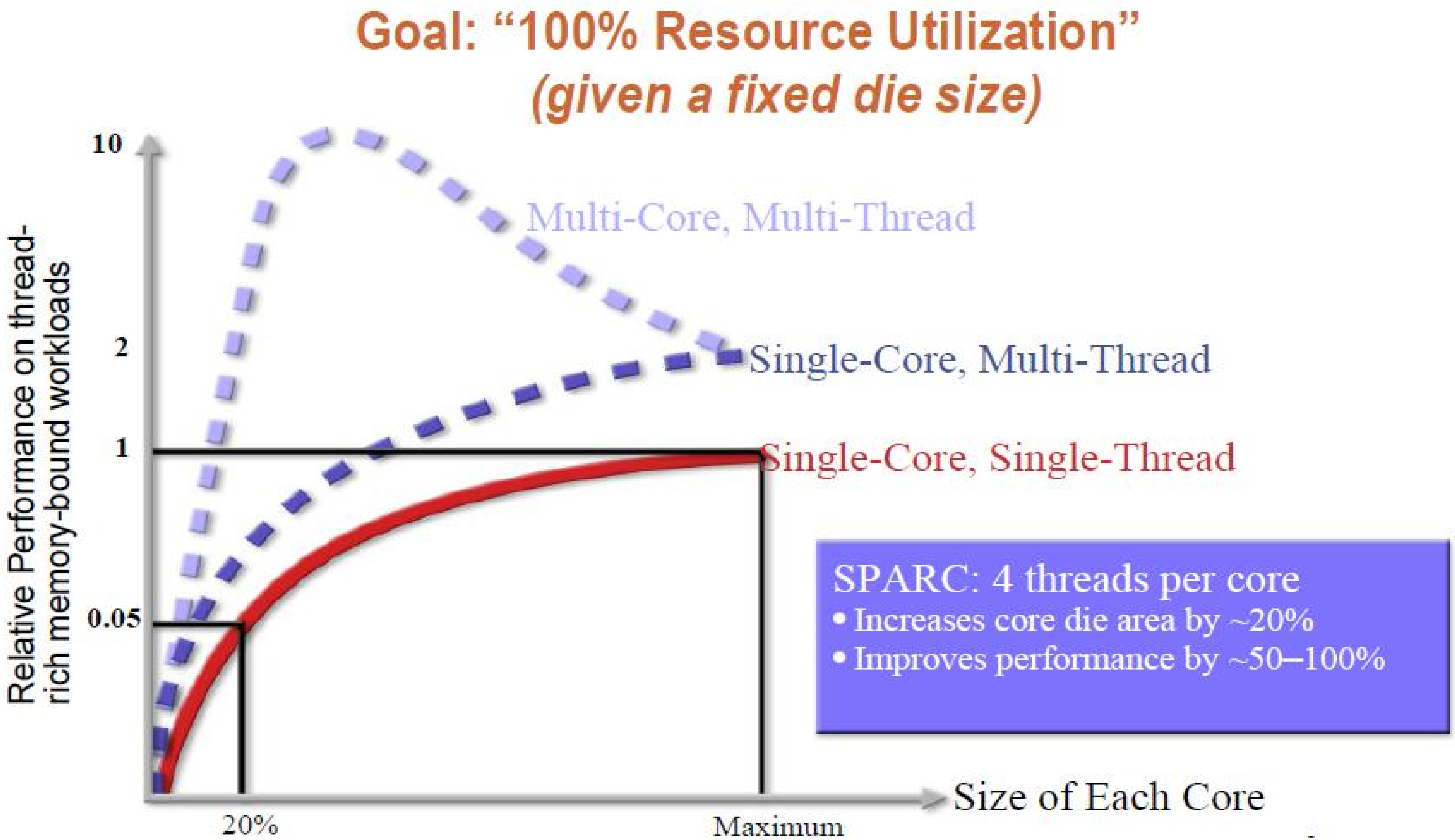


CMP
(Chip MultiProcessing, a.k.a. “multicore”)
n cores per processor

HMT
(Hardware Multithreading)
m threads per core

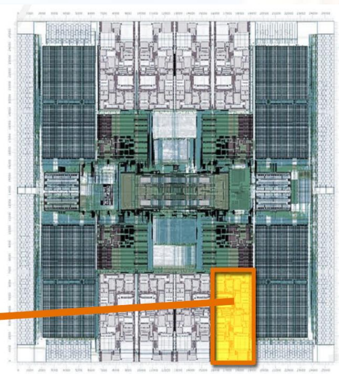
CMT
(Chip MultiThreading)
n x *m* threads per processor

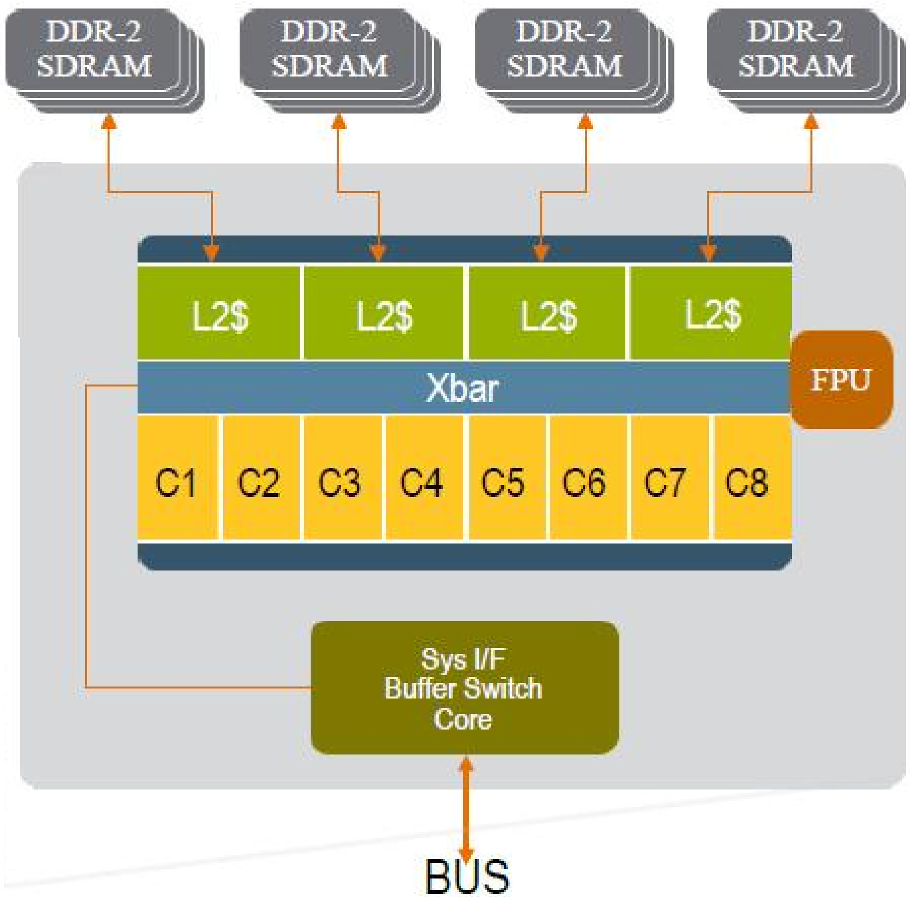
Why CMT works ?



T1 Architecture

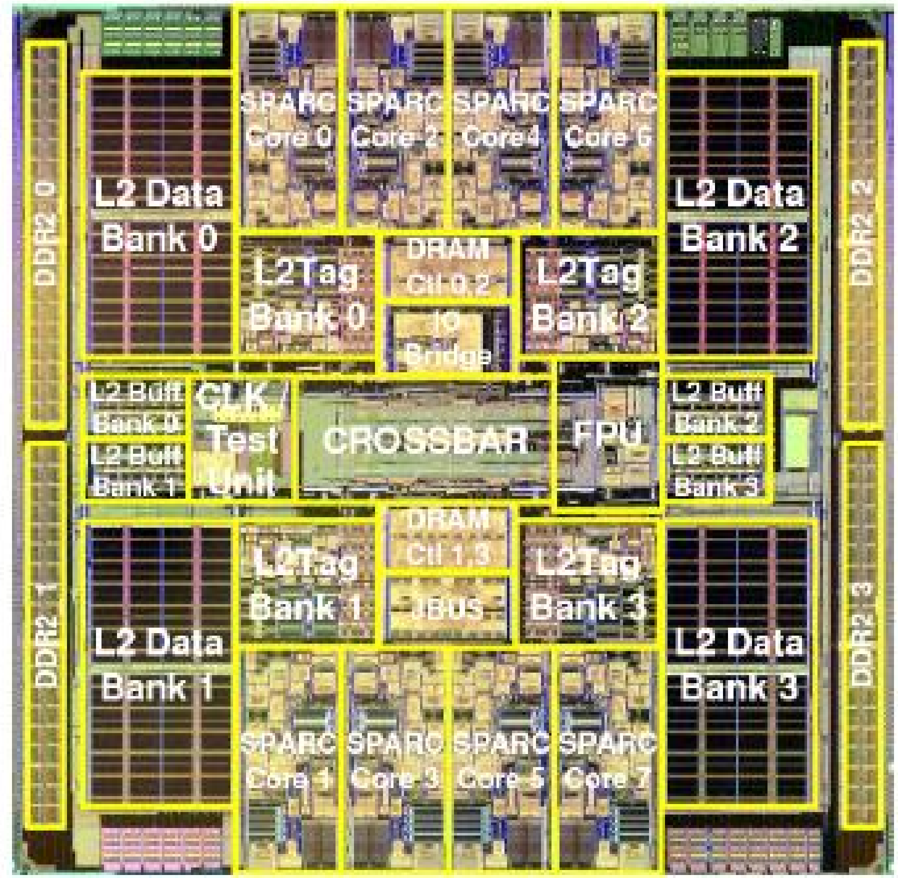
- 8 core/chip
- 4 Treads/core
- 1 FPU shared for 8 cores
- Core connected via crossbar switch(134.4GB/s)
- High-bandwidth, 12-way associative 3MB L2 cache
- 4 DDR2 channels (23GB/s)
- Power < 80W
- `300M transistors
- 378 mm² die





Chip Layout

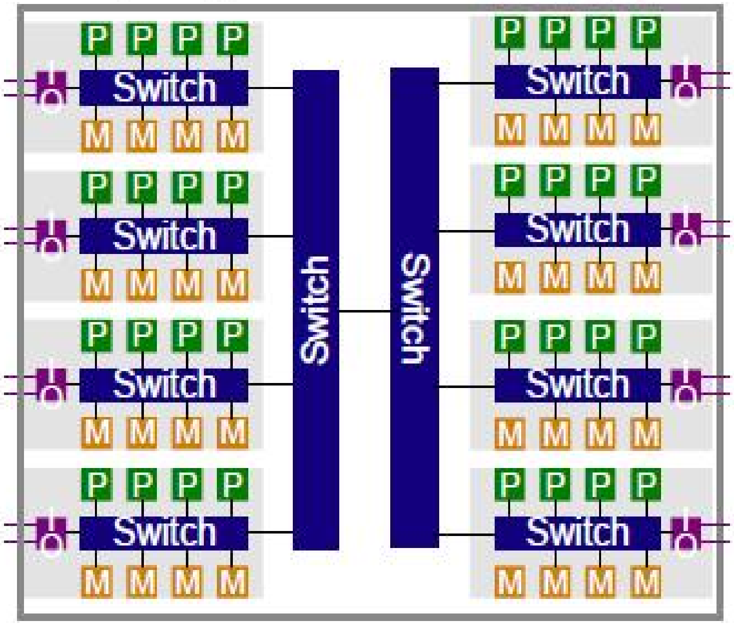
- L1 cache: 16KB Icache, 32B line; Dcache, 8KB, 16B line; 4-way set associative, 64B block, L1 miss penalty=23 cycles,
- L2 cache: 4 banks, each 750KB, 64B block, L2 miss penalty = 110 cycles
- L1 cache coherence is enforced by directory associate with each L2 cache block
- Memory controller on chip to decrease miss penalty



Compare with SMP

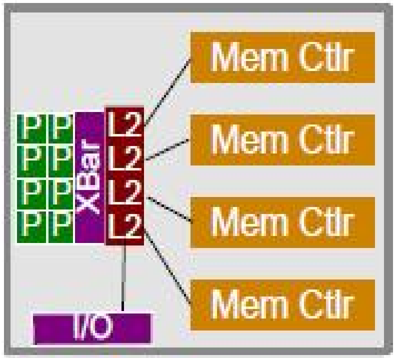
32-thread Traditional SMP System

Example: Typical SMP Machine Configuration



32-thread OpenSPARC T1 Processor

One motherboard, *no* switch ASICs



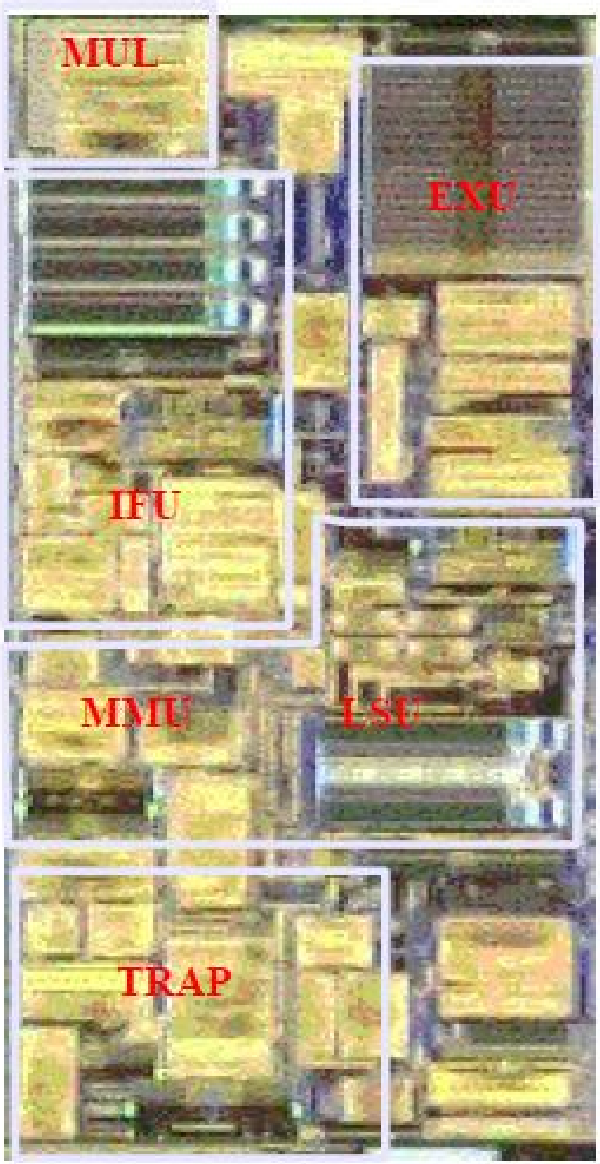
Direct crossbar interconnect

-- Lower cost
-- better RAS
-- lower BTUs,
-- lower and uniform latency,
-- greater and uniform bandwidth. . .

BTU: british theraml Unit

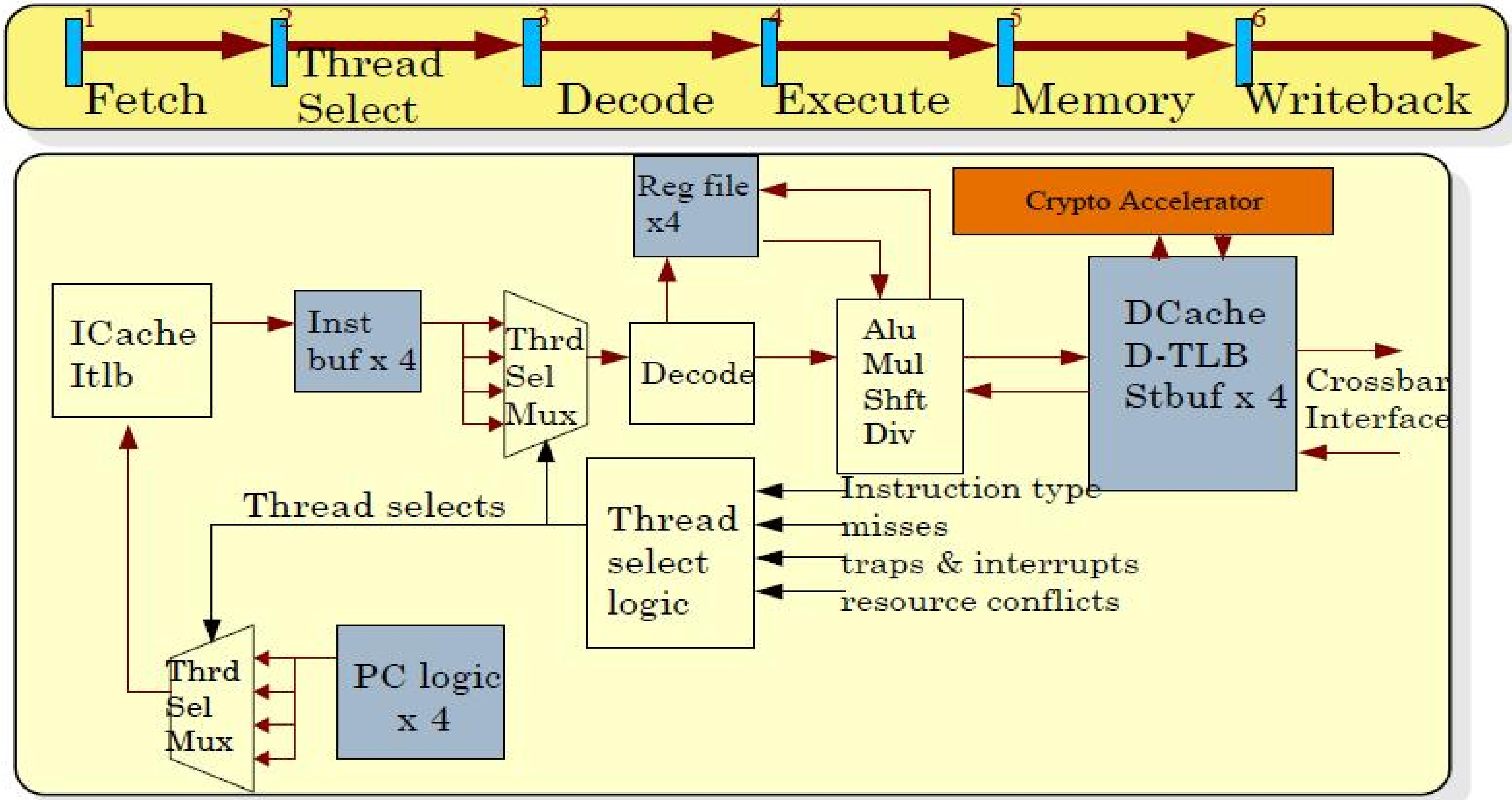
Core structure

- 4 threads /core
- Single issue
- 6 stage pipeline
- Unique resources/thread
  - Registers
  - Portions of I-fetch datapath
  - Store and Miss buffers
- Resource shared by 4 threads
  - Caches, TLBs, Exe Units
  - Pipeline registers and Data path
- Core area = 11mm^2 in 90nm
- MT adds ~20% area to core



ZHEJIANG UNIVERSITY 浙江大學

T1 processor core pipeline



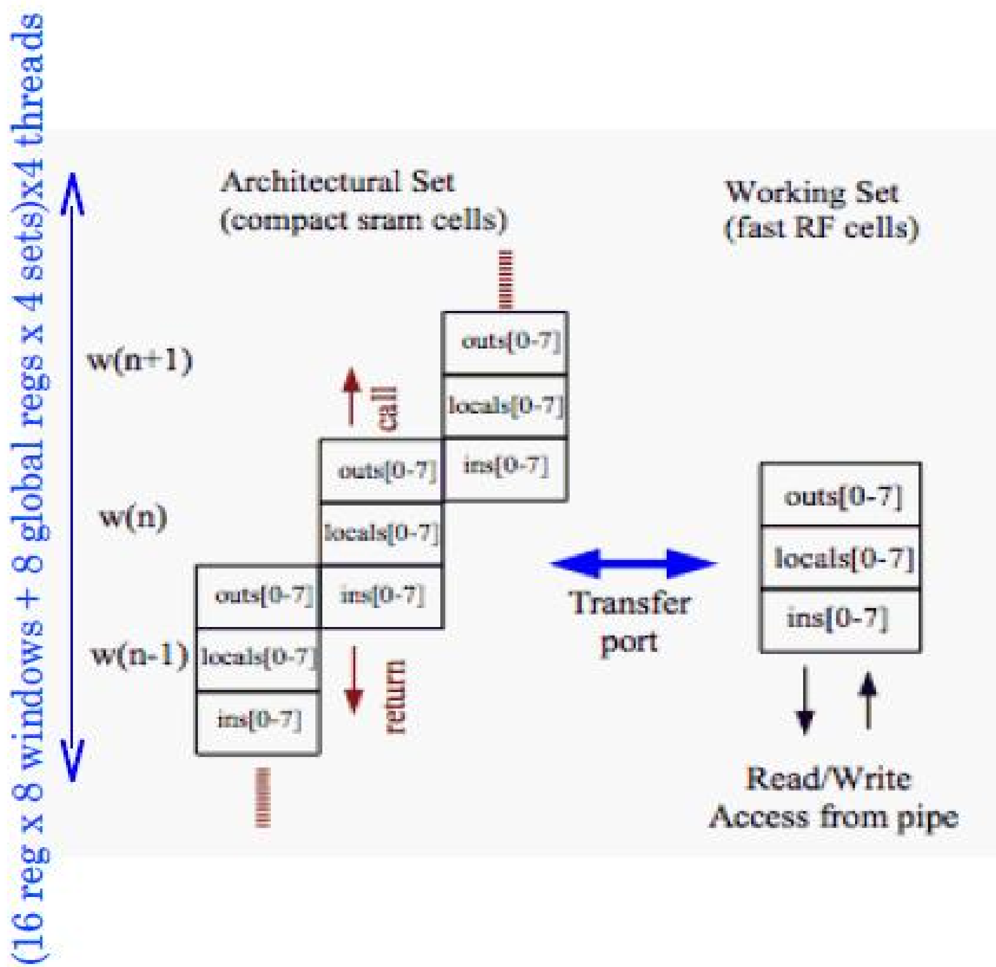
Thread Selection Policy

- Switch among available (ready to run) threads every cycle
  - Priority given to least-recently-executed thread
- Thread becomes not-ready-to-run due to:
  - Long latency operations like load, branch, mul, or div (load, branch incur 1 3-cycle delay)
  - Pipeline stall when cache miss, trap, or structure hazard

Instruction Fetch/Switch/Decode Unit (IFU)

- I-cache
  - 16KB data, 4ways, 32B line
  - Single-ported Instrctuion Tag
  - Dual-ported(1R/1W) Valid-bit array
  - Invalidate operations access Vilid-bit array, not instruction Tags
  - Pseudo-random replacement
- Fully associative Instruction TLB
  - 64 entries, Page sizes: 8K, 64K, 4M, 256M
  - Multiple hits in TLB prevented by doing auto-demap on fill
- 2 instructions fetched each cycle

Windowed Int Register file



- 5KB 3R/2W/1T structure
  - >640 64b regs
- Only the 32 registers from current window is visible to thread
- 8 global + 24 window register
- Window changing in background under thread switch
- Single cycle R/W accwss

ZHEJIANG UNIVERSITY

Execution Units

- Single ALU and shifter. ALU reused for branch address and virtual address calculation.
- Integer Multiplier
  - 5 clock latency, throughput of ½ per cycle
  - Contrains accumulate function for Mod Arithmetic
  - 1 integer mul allowed outstanding per core
  - Multipiier shared between Core Pipe and Modular Arithmetic unit on a round robinn basis
- Simple divider, with one divide outstanding per core
- Thread issuing a MUL/DIV will rollback and switch out if mul/div units is occcupied.

Load Store Unit (LSU)

- D-Cache complex
  - 8KB data, 4 ways, 16B line
  - Single ported Data Tag
  - Dual ported(1R/1W) Valid bit array
  - Pseudo-random replacement
  - Load are allocating, stores are non allocating
- 8 entry store buffer per thread, unified into single 32 entry array, with RAW bypassing

LSU(continued)

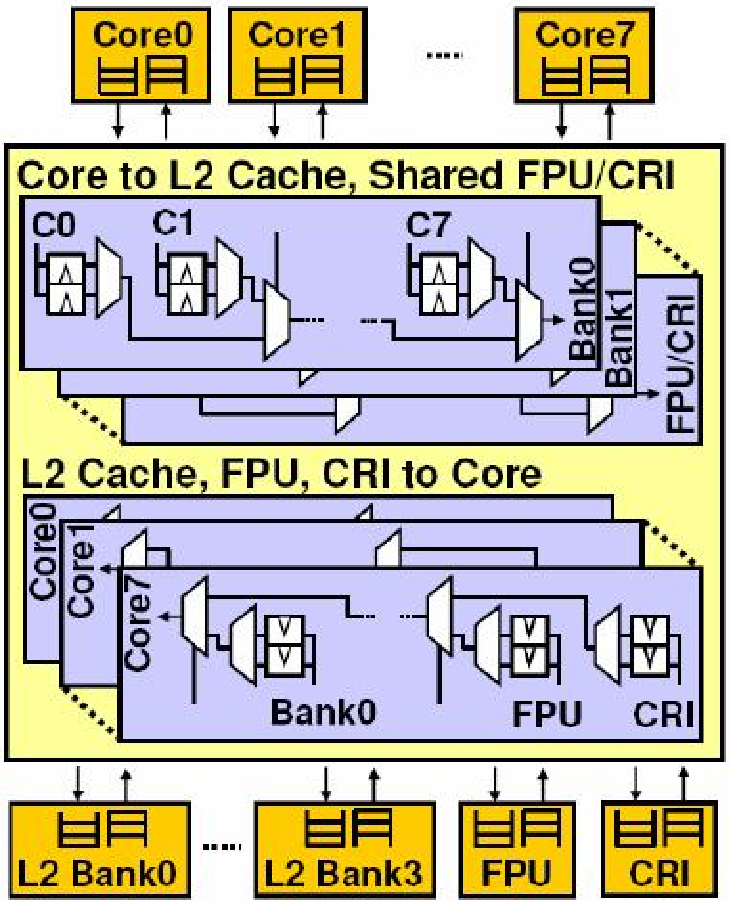
- Crossbar interface
  - LSU priorities requests to the crossbar for FPops, streaming ops, I and D misses, stores and interrupts.
- Request priority
  - Imiss>Idmiss>stores, {fpu, strm, interrupt}
  - Packed assembly for pcx.
- Handles return from crossbar and miantains order for cache updates and invalidtes.

ZHEJIANG UNIVERSITY

Other Functions

- Support for up to 64 pending interrupts per thread
- Support for 6 trap levels. Traps cause pipeline flush.
- Floating Point
  - The thread switch out when detecting ans FPop
  - Fpop is further decoded and FRF is read
  - FPop with operands are packetized and shipped over the crossbar to the FPU
  - FPU returned the result via crossbar
  - Write back to FRF and thread restart

Cross Bar



- Each requestor queues up to 2 packets per destination
- 3 stage pipeline: request, arbitrate and transmit
- Oldest requestor first
- Core to cache bus;
  - Address+double word store
- Cache to core bus
  - 16B line 32B Icache line fill delivered in 2 back to back clock cycles

L2 Cache

- 3MB, 4-way banked, 12-way associative, writeback
- 64B line size, 64B interleaved between banks
- Latency: 8 clocks for load, 9 clocks for I-miss, critical chunk first
- 16 outstanding misses per bank → 64 total
- Coherence maintained by shadowing L1 tags in an L2 directory structure
- L2 is global visibility. DMA from IO is serialised write traffic from cores in L2
- L2 has a 8-stage pipeline (C1-C8)

L2 Cache --Directory

- Directory shadows L1 tags
  - 2048 entries, half-half for Icache and Dcache
  - L1 set index and L2 bank interleaving is such that ¼ of L1 entries come from each L2 bank
  - On and L1 miss, the L1 replacement way and set index identify the physical location of tag
  - On a store, write invalidate will be done
    - Invalidate operation are pointer to the physical location of L1. (no tag lookup in L1)
    - An acknowledgement packet is sent to the request CPU, an invalidation packet sent to all other core

L2 Cache Single Bank

- Arbiter (instruction from CCX, DMA instruction, stall signals from pipeline, instruction from FB and MB)
- L2 tag
- L2 VUAD array –Valid, Used, Allocated(set when a line is picked for replacement) Dirty
- L2 data
- Input Queue (16 entry FIFO) – packets arriving on the PCX
- Output Queue (16 entry FIFO) – operations waiting for access to CPX(cache-to-processor interface)
- Snoop input Queue (2-entry FIFO)
- Miss buffer (MB) (L2 misses, same line address with previous miss or write-back buffer, atomics and partial stores etc)
- Fill buffre (temp store data from DRAM)
- Write-back buffer
- Remote DMA write buffer (4 entry) for DMA write
- L2 cache directory –coherency management and maintain inclusive property

Coherence/consistency

- Load update directory & fill the L1 on return
- Stores are non-allocating in L1
  - TSO, PSO, RMO (one TSO store outstanding to L2 per thread)
  - Check directory and determine L1 hit
  - Directory send store ack/inv to core
  - Store update happens to Dcache on store ack
- Crossbar orders responses across cache banks
- 3 atomic instructions are processed by L2 cache
  - LDSTUB, SWAP, Compare and swap(CAS)
  - Two passes downto cache pipeline

On chip Mem controller

- 4 independent DDRII DRAM channels
- Upto 128GB memory size
- 25GB/s peak bandwidth
- Schedules across 8 rds + 8 writes
- 128+16b interface, chipkill support, correction, byte error detection
- Designed to work from 125-200Mhz

T1 Performance

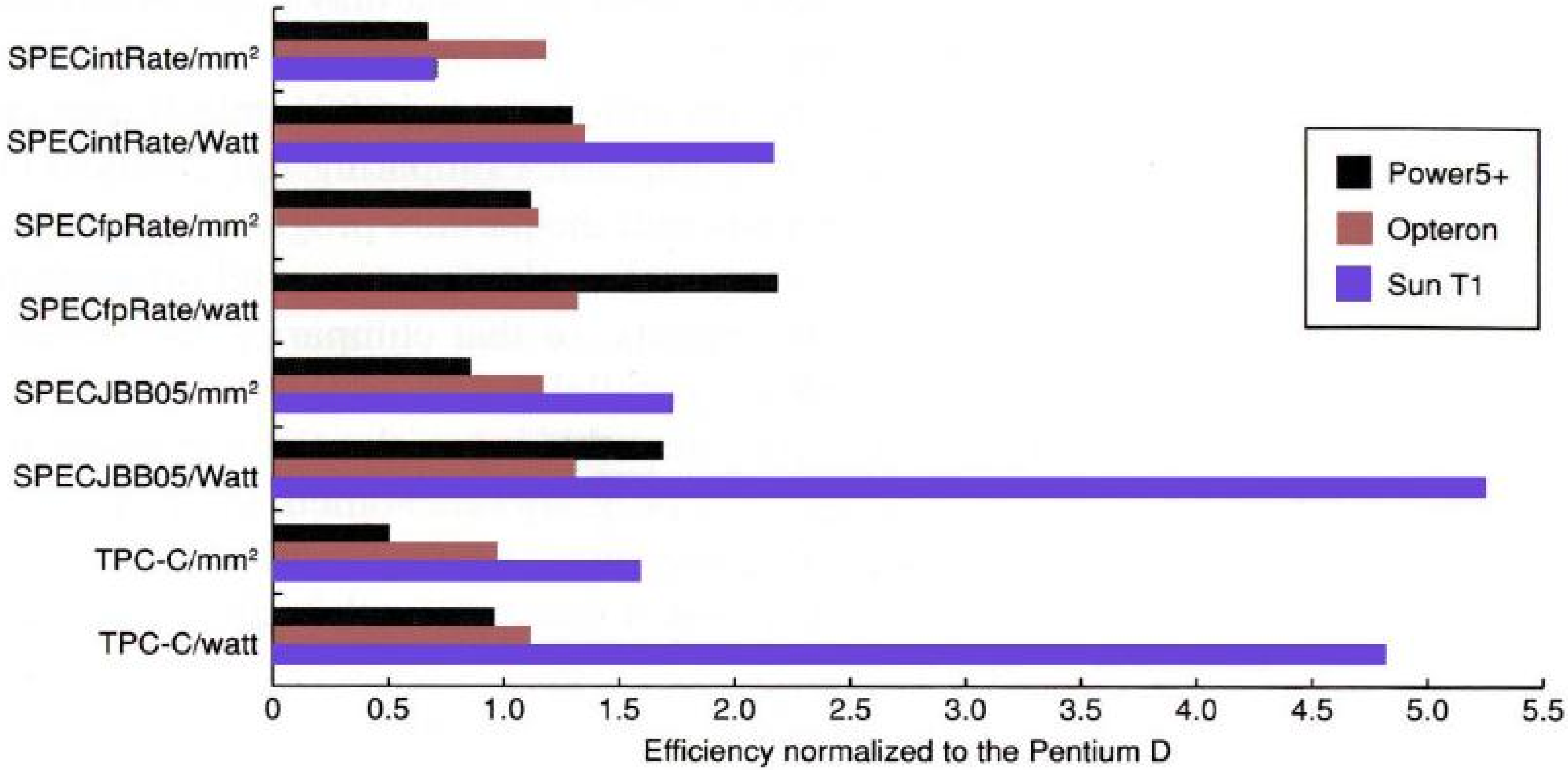


Figure 4.34 Performance efficiency on SPECRate for four dual-core processors, normalized to the Pentium D metric (which is always 1).

Conclusion

- Sun T1 provides much better performance in terms of performance/watt, especially for TPC-C-like and SPECJBB05 benchmark.
- TLP approach is much more power efficient than an ILP-intensive approach for multithreaded applications.
- CMT may provide a way to increase performance in a power efficient fashion.