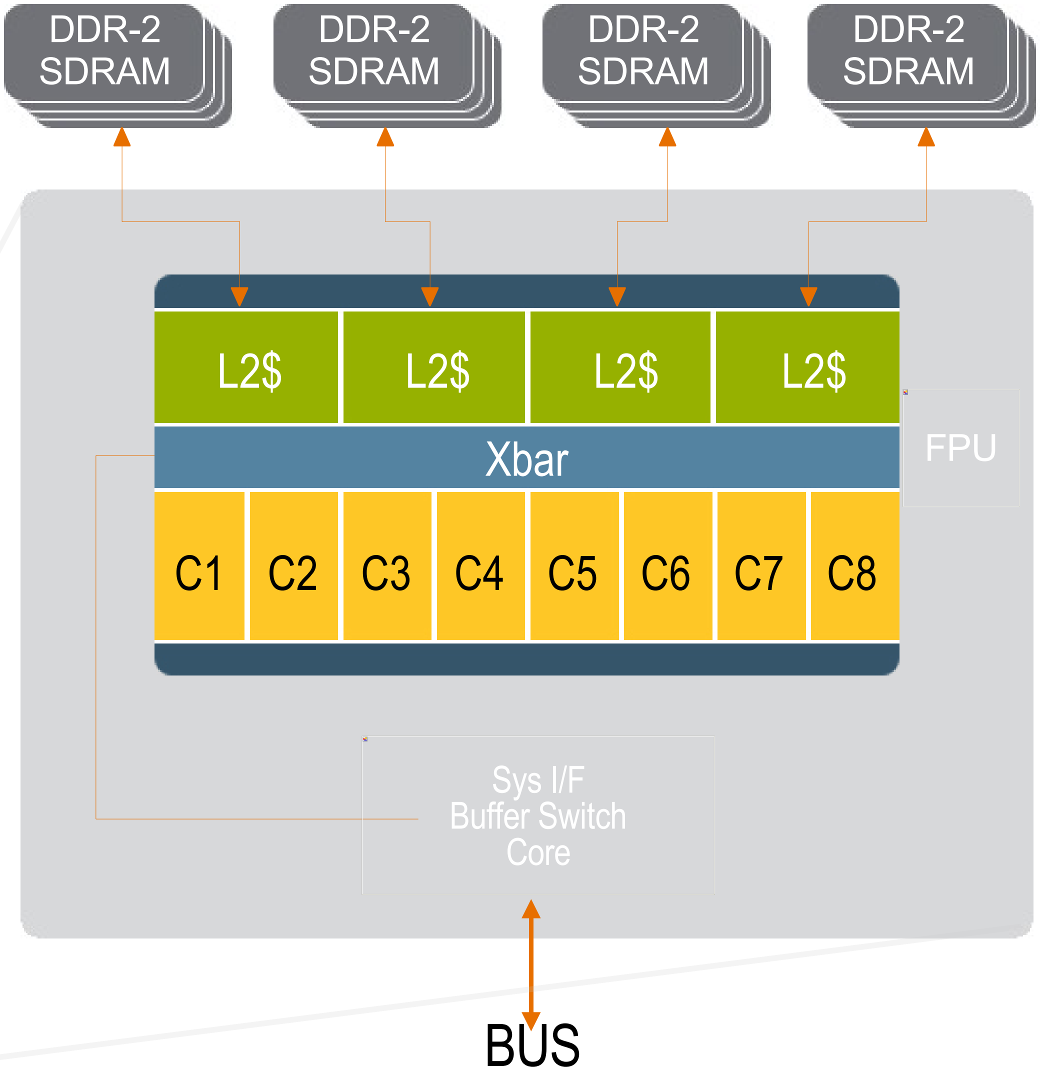
Chapter Four

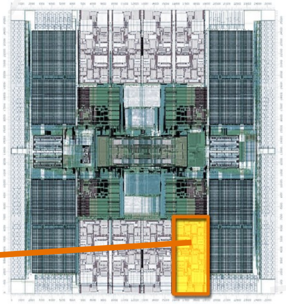
OPENSPARC T1 OVERVIEW

Denis Sheahan
Distinguished Engineer
Niagara Architecture Group
Sun Microsystems

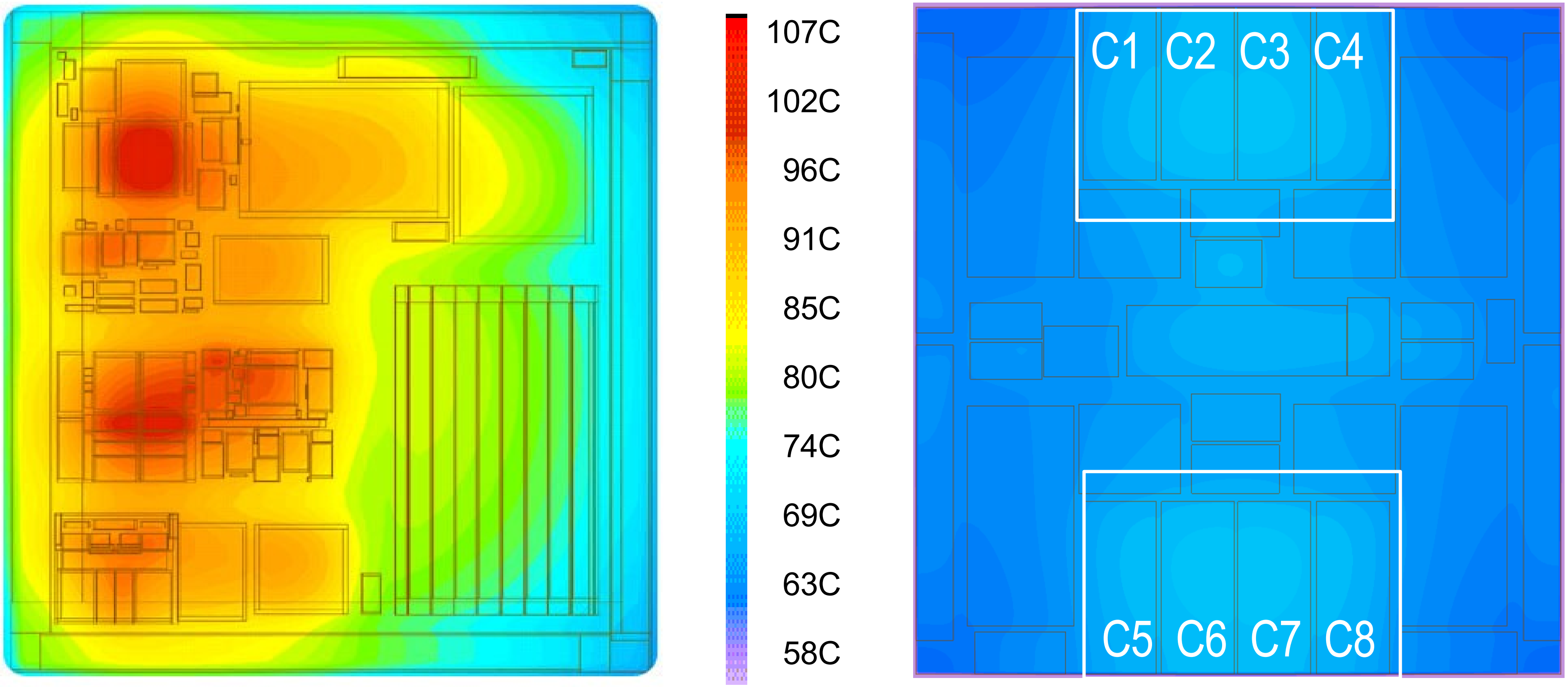
UltraSPARC T1 Processor

- SPARC V9 (Level 1) implementation
- Up to eight 4-threaded cores (32 simultaneous threads)
- All cores connected through high bandwidth (134.4GB/s) crossbar switch
- High-bandwidth, 12-way associative 3MB Level-2 cache on chip
- 4 DDR2 channels (23GB/s)
- Power : < 80W
- ~300M transistors
- 378 sq. mm die





Faster Can Be Cooler



Single-Core Processor

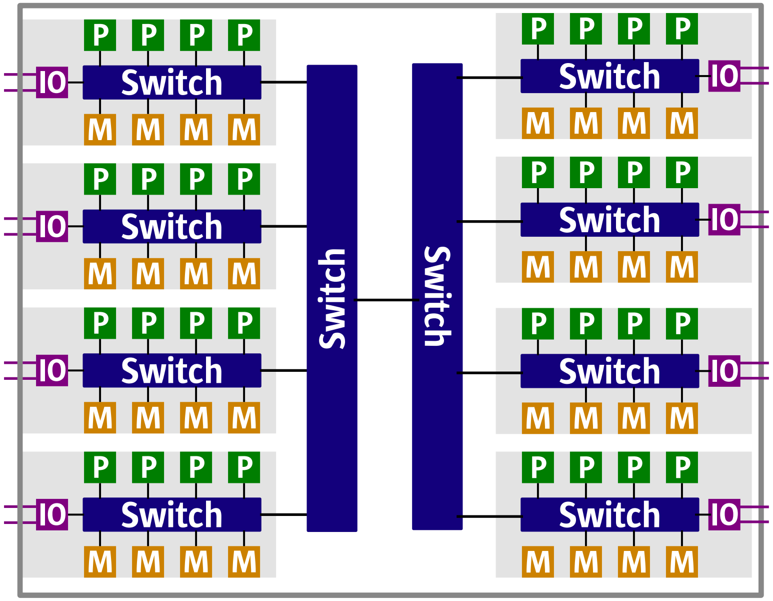
CMT Processor

(Not to Scale)

CMT: On-chip = High Bandwidth

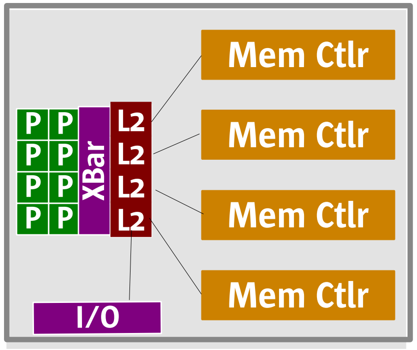
32-thread Traditional SMP System

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. . .

CMT Benefits

↑ Performance

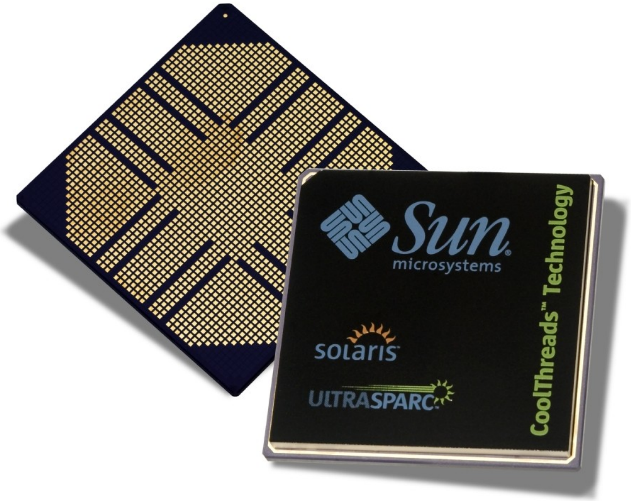
↓ Cost

- Fewer servers
- Less floor space
- Reduced power consumption
- Less air conditioning
- Lower administration and maintenance

↑ Reliability

CMT Pays Off with CoolThreads™ Technology

Sun Fire T1000



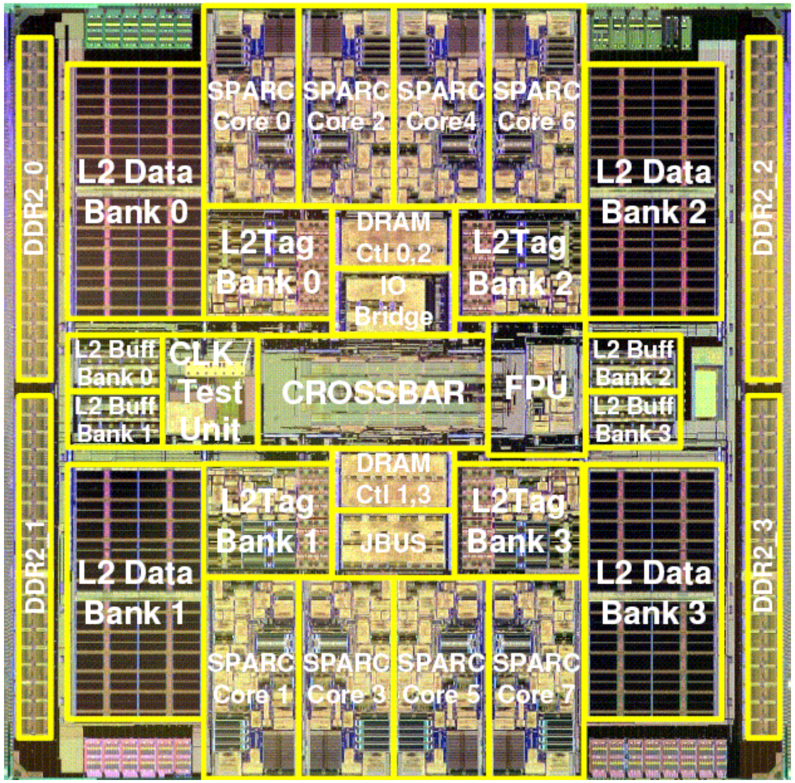
Sun Fire T2000

- Up to 5x the performance
- As low as 1/5 the energy
- As small as 1/4 the size

InfoWorld 2006 TECHNOLOGY OF THE YEAR AWARD

*See disclosures

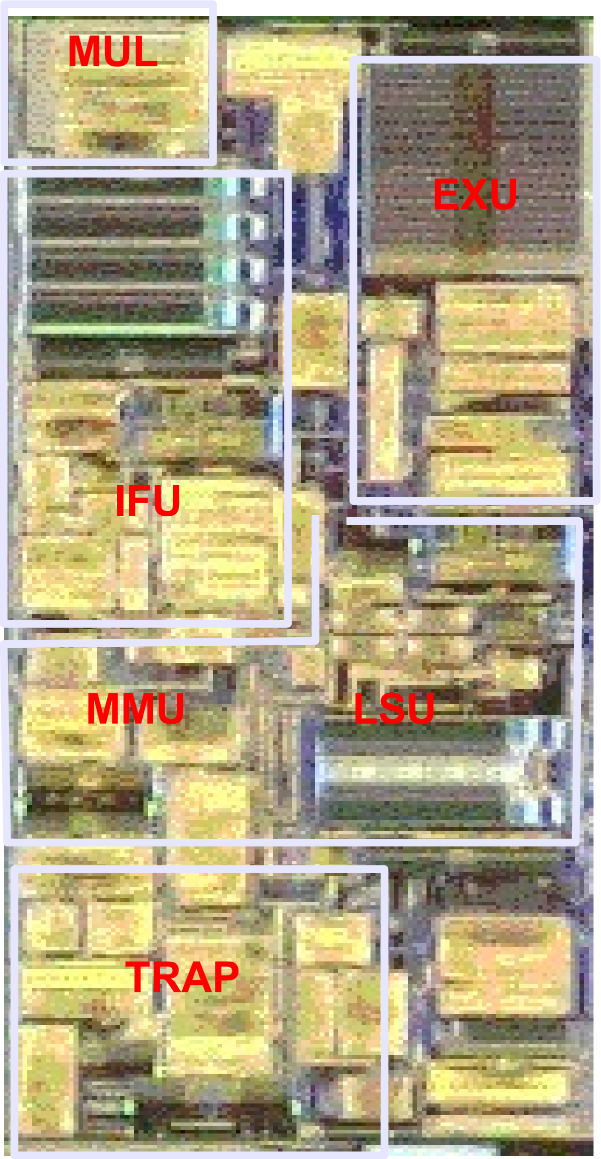
UltraSPARC-T1: Choices & Benefits



- Simple core (6-stage, only 11mm^2 in 90nm), 1 FPU
  → maximum # of cores/threads on die
  → pipeline built from scratch, useful for multiple generations
  → modular, flexible design ... ***scalable*** (up and down)
- Caches, DRAM channels shared across cores
  → better area utilization
- Shared L2 cache
  → cost of coherence misses decrease by order of magnitude
  → enables highly efficient multi-threaded software
- On-die memory controllers
  → reduce miss latency
- Crossbar switch
  → good for b/w, latency, functional verification

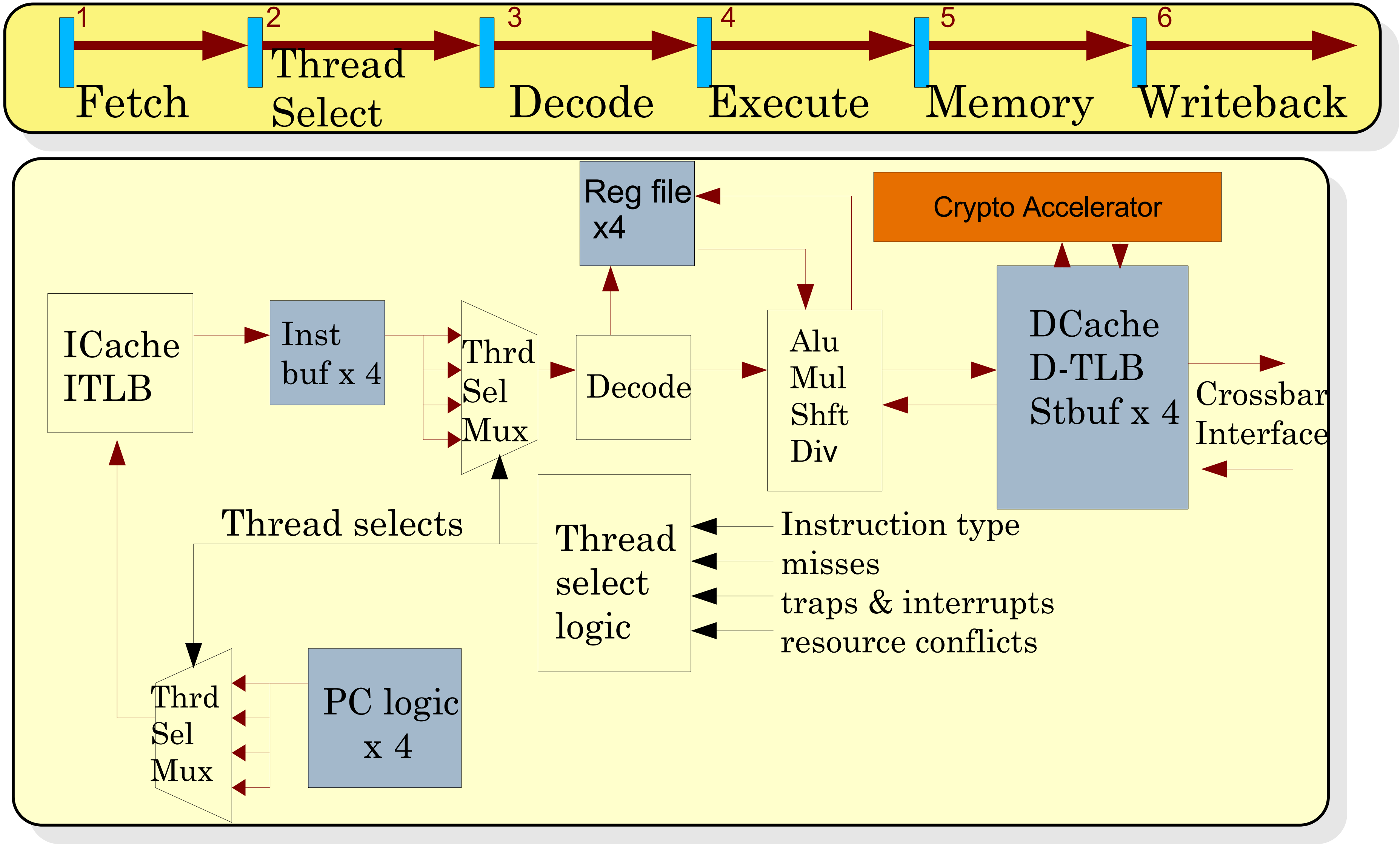
For reference: in 90nm technology, included 8 cores, 32 threads, and only dissipate 70 W

UltraSPARC-T1 Processor Core



- Four threads per core
- Single issue 6 stage pipeline
- 16KB I-Cache, 8KB D-Cache
- > Unique resources per thread
  - > Registers
  - > Portions of I-fetch datapath
  - > Store and Miss buffers
- > Resources shared by 4 threads
  - > Caches, TLBs, Execution Units
  - > Pipeline registers and DP
- Core Area = 11mm^2 in 90nm
- MT adds ~20% area to core

UltraSPARC T1 Processor Core Pipeline



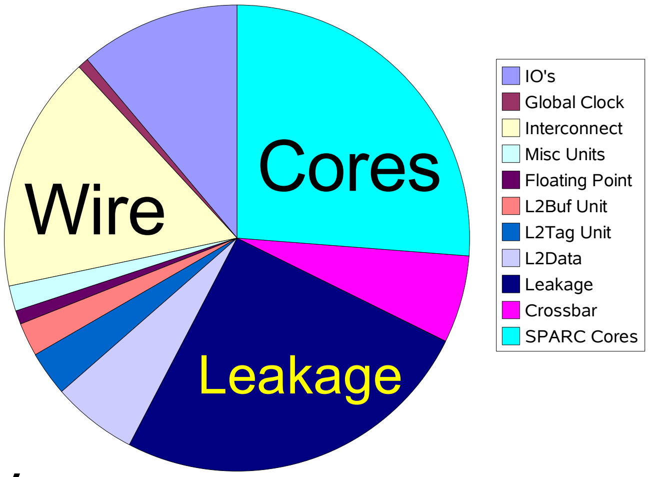
...blue units are replicated *per thread* on core

Thread Selection Policy

- Every cycle, switch among available (ready to run) threads
  - priority given to least-recently-executed thread
- Thread becomes not-ready-to-run due to:
  - Long latency operations like load, branch, mul, or div
  - Pipeline stall such as cache miss, trap, or resource conflict
- Loads are speculated as cache hits, and the thread is switched in with lower priority

OpenSPARC T1 Power

- Power Efficient Architecture
  - Single issue, in-order six stage pipeline
  - Minimal speculation, predication or branch prediction
- Thermal monitoring for power throttling
  - 3 external power throttle pins
    - Controlled by thermal diodes
    - Stall cycles injected, affecting all threads
  - Memory throttling
- Design Implementation
  - Fully static design
  - Fine granularity clock gating
    - Limited clock issue on stall, FGU
    - Limited L2 Cache & Memory clock gating
  - Wire classes optimized for power x delay

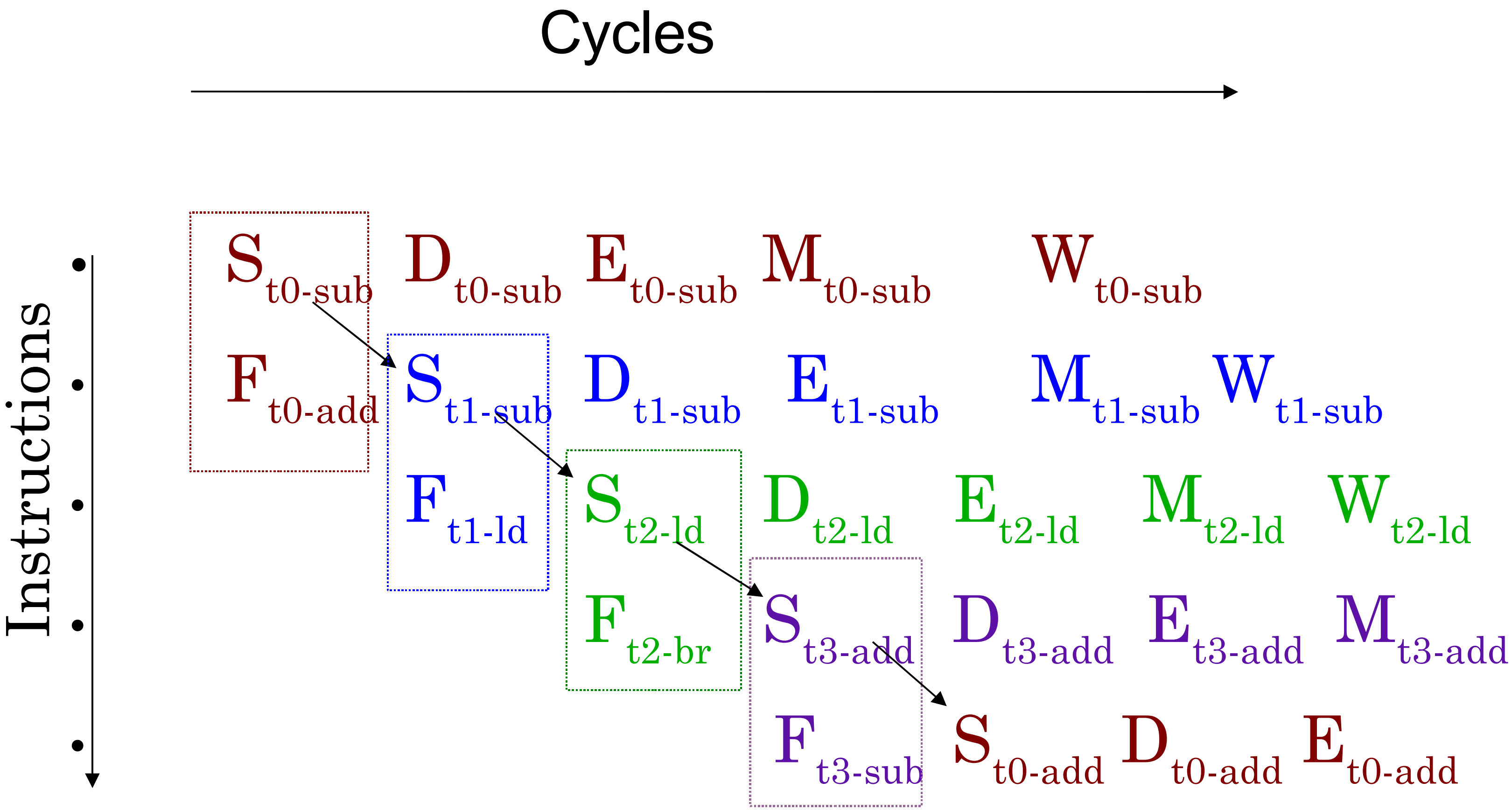


T1 Power Components

OpenSPARC T1 Microarchitecture

OpenSPARC
October 2008

Thread Selection – All Threads Ready


Cycles
Instructions
$S_{\text{t0-sub}}$ $D_{\text{t0-sub}}$ $E_{\text{t0-sub}}$ $M_{\text{t0-sub}}$ $W_{\text{t0-sub}}$
$F_{\text{t0-add}}$ $S_{\text{t1-sub}}$ $D_{\text{t1-sub}}$ $E_{\text{t1-sub}}$ $M_{\text{t1-sub}}$ $W_{\text{t1-sub}}$
$F_{\text{t1-ld}}$ $S_{\text{t2-ld}}$ $D_{\text{t2-ld}}$ $E_{\text{t2-ld}}$ $M_{\text{t2-ld}}$ $W_{\text{t2-ld}}$
$F_{\text{t2-br}}$ $S_{\text{t3-add}}$ $D_{\text{t3-add}}$ $E_{\text{t3-add}}$ $M_{\text{t3-add}}$
$F_{\text{t3-sub}}$ $S_{\text{t0-add}}$ $D_{\text{t0-add}}$ $E_{\text{t0-add}}$

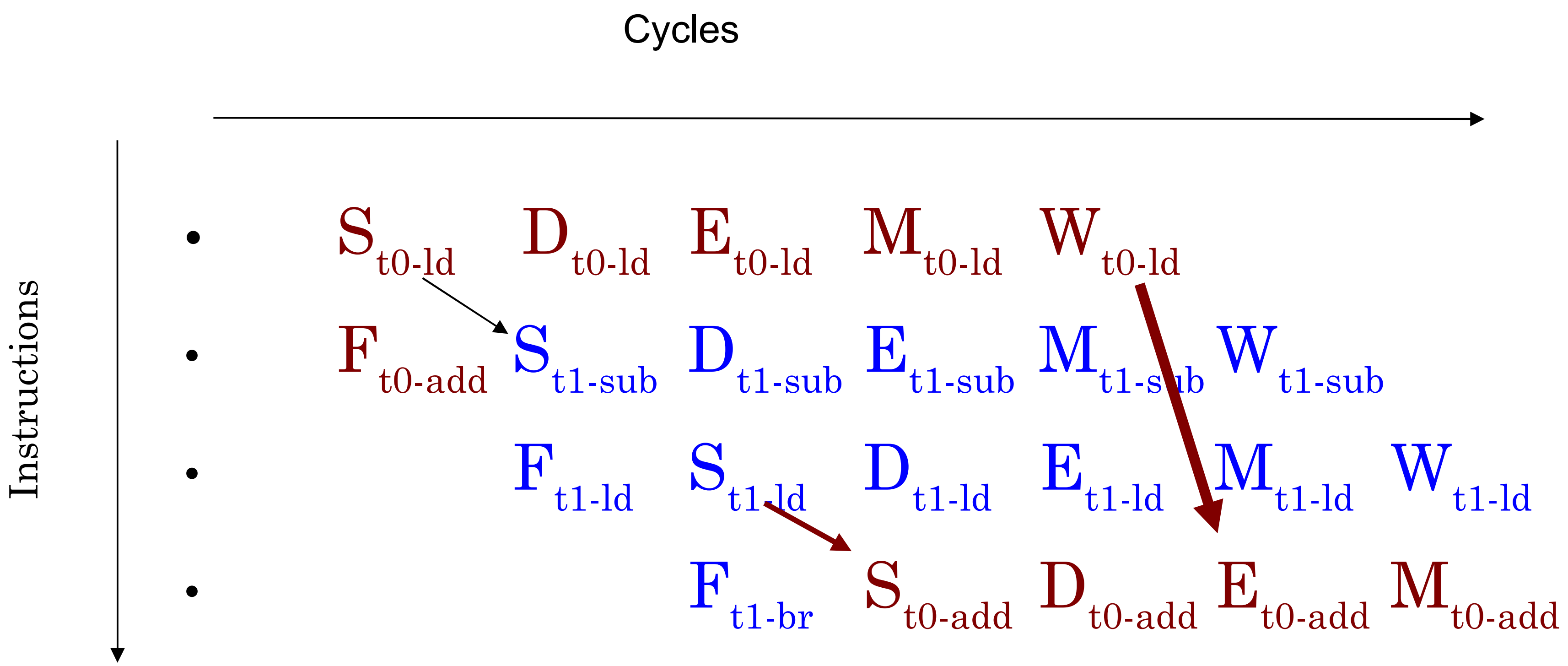
Thread 0:
sub
add

Thread 1:
sub
ld

Thread2:
ld
br

Thread 3:
add
sub

Thread Selection – Two Threads Ready



Cycles

Instructions

- $S_{\text{t0-ld}}$ $D_{\text{t0-ld}}$ $E_{\text{t0-ld}}$ $M_{\text{t0-ld}}$ $W_{\text{t0-ld}}$
- $F_{\text{t0-add}}$ $S_{\text{t1-sub}}$ $D_{\text{t1-sub}}$ $E_{\text{t1-sub}}$ $M_{\text{t1-sub}}$ $W_{\text{t1-sub}}$
- $F_{\text{t1-ld}}$ $S_{\text{t1-ld}}$ $D_{\text{t1-ld}}$ $E_{\text{t1-ld}}$ $M_{\text{t1-ld}}$ $W_{\text{t1-ld}}$
- $F_{\text{t1-br}}$ $S_{\text{t0-add}}$ $D_{\text{t0-add}}$ $E_{\text{t0-add}}$ $M_{\text{t0-add}}$

Thread 0:	Thread 1:
ld	sub
add	ld
	br

Thread '0' is speculatively switched in before cache hit information is available, in time for the 'load' to bypass data to the 'add'

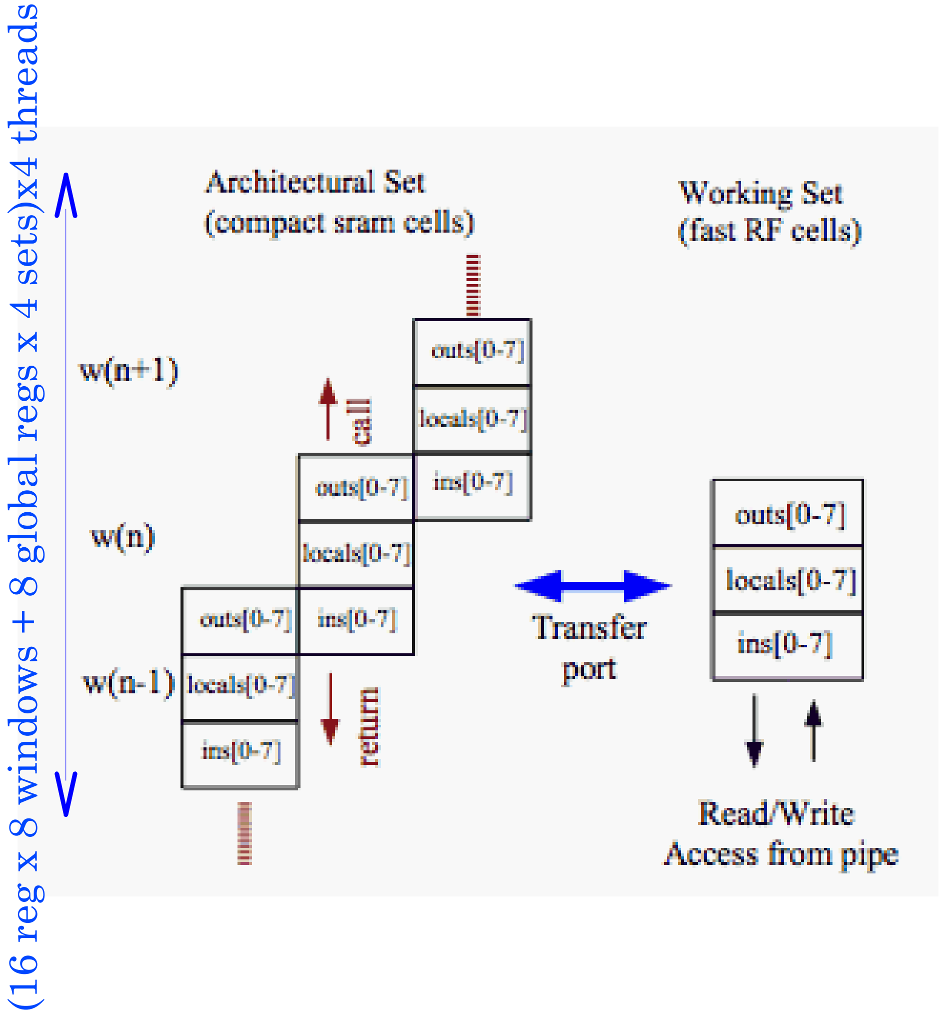
Instruction Fetch/Switch/Decode Unit(IFU)

- I-cache
  - 16KB data, 4ways, 32B line size
  - Single-ported Instruction Tag
  - Dual-ported(1R/1W) Valid-bit array to hold Cache line state of valid/invalid
  - Invalidate operations access Valid-bit array, not Instruction Tags
  - Pseudo-random replacement
- Fully Associative Instruction TLB
  - 64 entries, Page sizes: 8k, 64k, 4M, 256M
  - Pseudo LRU replacement.
  - Multiple hits in TLB prevented by doing auto-demap on fill

IFU Functions (Cont'd)

- 2 instructions fetched each cycle, though only one is issued/clk. Reduces I$ activity and allows opportunistic line fill.
- 1 outstanding miss/thread, and 4 per core. Duplicate misses do not request to L2
- PC's, NPC's for all live instructions in machine maintained in IFU

Windowed Integer Register File



- 5KB 3R/2W/1T structure
  - 640 64-bit regs with ECC!
- Only the 32 registers from current window is visible to thread
- Window changing in background under thread switch. Other threads continue to access IRF
- Compact design with 6T cells for architectural set & multi ported cell for working set.
- Single cycle R/W access

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 for area saving
  - Contains accumulate function for Mod Arithmetic.
  - 1 integer mul allowed outstanding per core.
  - Multiplier shared between Core Pipe and Modular Arithmetic unit on a round robin basis.
- Simple non restoring divider, with one divide outstanding per core.
- Thread issuing a MUL/DIV will rollback and switch out if another thread is occupying the mul/div units.

Load Store Unit(LSU)

- D-Cache complex
  - 8KB data, 4ways, 16B line size
  - Single ported Data Tag.
  - Dual ported(1R/1W) Valid bit array to hold Cache line state of valid/invalid
  - Invalidates access Vbit array but not Data Tag
  - Pseudo-random replacement
  - Loads are allocating, stores are non allocating.
- DTLB: common macro to ITLB(64 entry FA)
- 8 entry store buffer per thread, unified into single 32 entry array, with RAW bypassing.

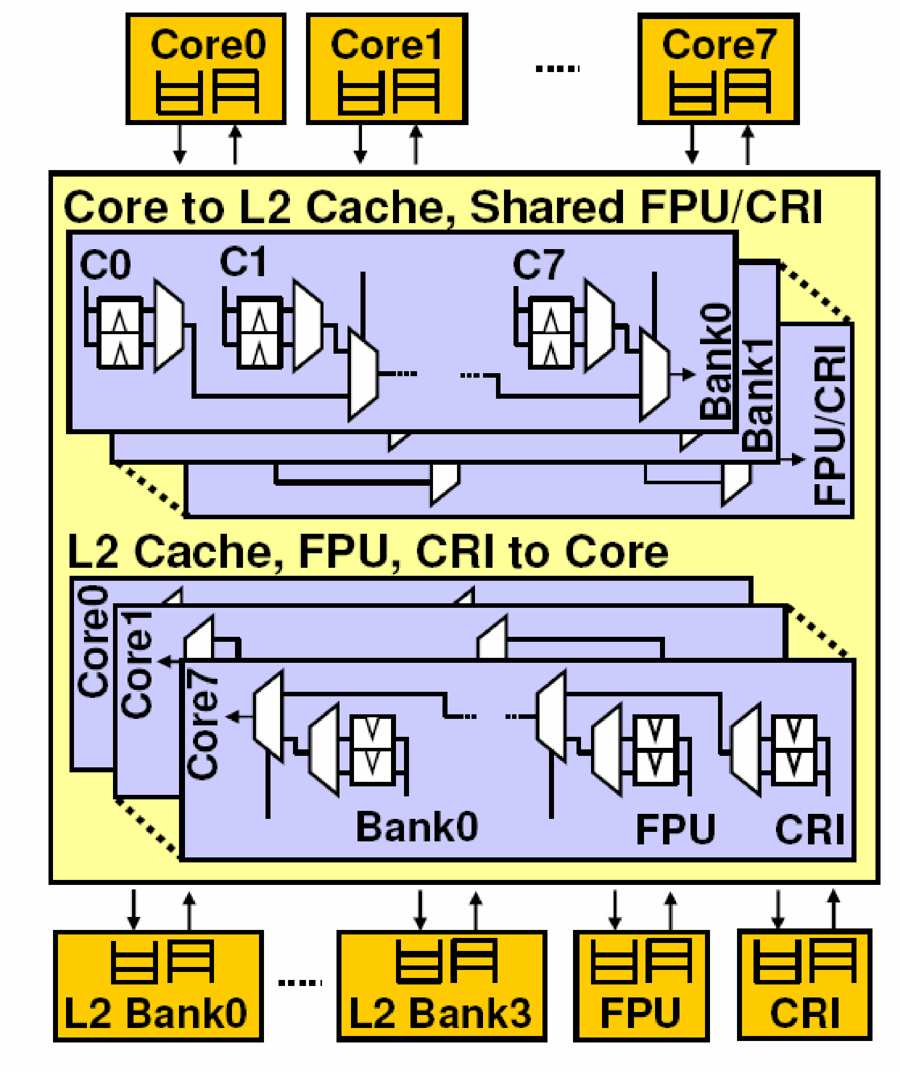
LSU(Cont'd)

- Single load per thread outstanding. Duplicate request for the same line not sent to L2
- Crossbar interface
  - LSU prioritizes requests to the crossbar for FPops, Streaming ops, I and D misses, stores and interrupts etc.
  - Request priority: imiss>ldmiss>stores, {fpu,strm,interrupt}.
  - Packet assembly for pcx.
- Handles returns from crossbar and maintains order for cache updates and invalidates.

Other Functions

- Support for 6 trap levels. Traps cause pipeline flush and thread switch until trap PC is available
- Support for up to 64 pending interrupts per thread
- Floating Point
  - FP registers and decode located within core
  - On detecting an Fpop
    - The thread switches out
    - Fpop is further decoded and FRF is read
    - Fpop with operands are packetized and shipped over the crossbar to the FPU
    - Computation done in FPU and result returned via crossbar
    - Writeback completed to FRF (floating-point register file) and thread restart

Crossbar



- Each requestor queues up to 2 packets per destination.
- 3 stage pipeline: Request, Arbitrate and Transmit
- Centralized arbitration with oldest requestor getting priority
- Core to cache bus optimized for address + doubleword store
- Cache to core bus optimized for 16B line fill. 32B I$ line fill delivered in 2 back to back clks

L2 Cache

- 3MB, 4-way banked, 12-way SA, Writeback
- 64B line size, 64B interleaved between banks
- Pipeline latency: 8 clks for Load, 9 clks for I-miss, with critical chunk returned first
- 16 outstanding misses per bank -> 64 total
- Coherence maintained by shadowing L1 tags in an L2 directory structure.
- L2 is point of global visibility. DMA from IO is serialized wrt traffic from cores in L2

L2 Cache – Directory

- Directory shadows L1 tags
  - L1 set index and L2 bank interleaving is such that ¼ of L1 entries come from each L2 bank
  - On an L1 miss, the L1 replacement way and set index identify the physical location of the tag which will be updated by miss address
  - On a store, directory will be CAM'ed
    - Directory entries collated by set so only 64 entries need to be CAM'ed .
      Scheme is quite power efficient.
    - Invalidate operations are a pointer to the physical location in the L1, eliminating the need for a tag lookup in L1.

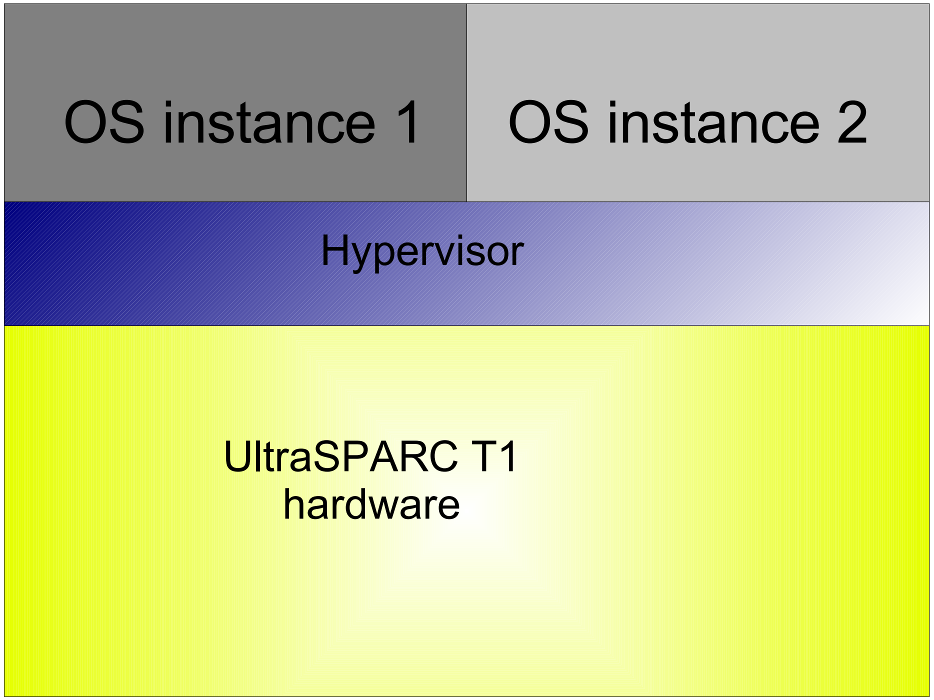
Coherence/Ordering

- Loads update directory & fill the L1 on return
- Stores are non-allocating in L1
  - Two flavors of stores: TSO, RMO.
    One TSO store outstanding to L2 per thread to preserve store ordering. No such limitation on RMO stores
  - No tag check done at store buffer insert
  - Stores check directory and determine L1 hit.
  - Directory sends store ack/inv to core
  - Store update happens to D$ on store ack
- Crossbar orders responses across cache banks

On-Chip Memory Controller

- 4 independent DDR2 DRAM channels
- Can supports memory size of upto 128GB
- 25GB/s peak bandwidth
- Schedules across 8 reads + 8 writes
- Can be programmed to 2 channel mode in reduced configuration (reduced corresponding cores)
- 128+16b interface, chip-kill support, nibble error correction, byte error detection
- Designed to work from 125-200Mhz

Virtualization



- Hypervisor layer virtualizes CPU
- Multiple OS instances
- Better RAS: failures in one domain do not affect other domains
- Improved OS portability to newer hardware

Virtualization on UltraSPARC T1

- Implementation on UltraSPARC-T1
  - Hypervisor uses Physical Addresses
  - Supervisor* sees 'Real Addresses' – a PA abstraction
  - VA translated to RA, and then to PA.
    Niagara(T1) MMU and TLB provides h/w support.
  - Up to 8 partitions can be supported.
    3-bit partition ID is part of TLB translation checks
  - Additional trap level added for hypervisor use

* supervisor = privileged-mode software = operating system
(for example, Solaris, Linux, *BSD, ...)

OpenSPARC Slide-Cast

In 12 Chapters

Presented by OpenSPARC designers, developers, and programmers

- to guide users as they develop their own OpenSPARC designs and
- to assist professors as they teach the next generation