



OpenSPARC™

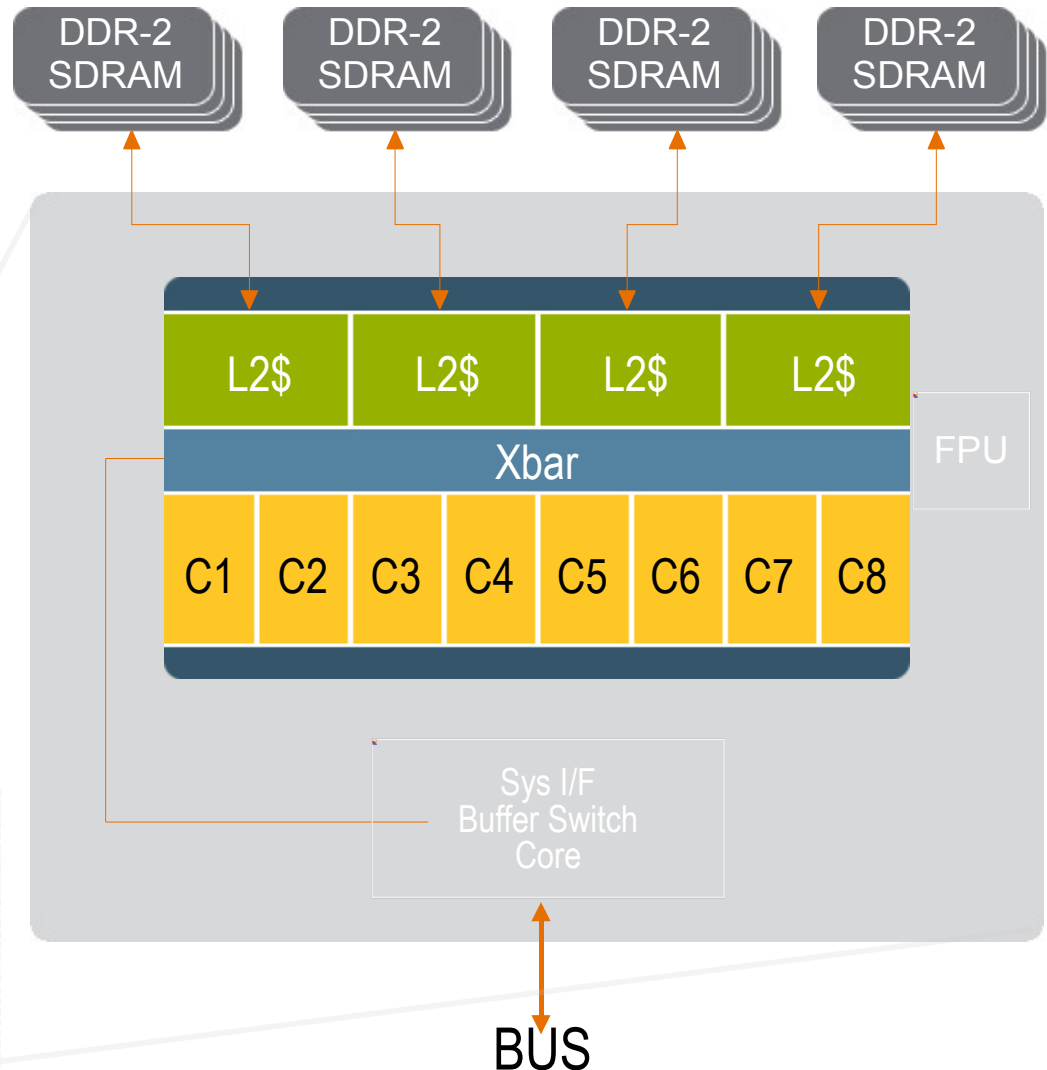
OPENSPARC T1 & T2 OVERVIEW

Rick Hetherington
Chief Technology Officer
Microelectronics
Sun Microsystems

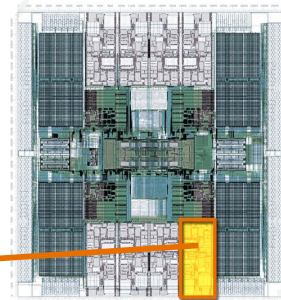


UltraSPARC T1 Processor

- 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

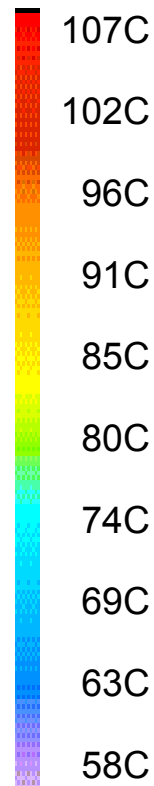
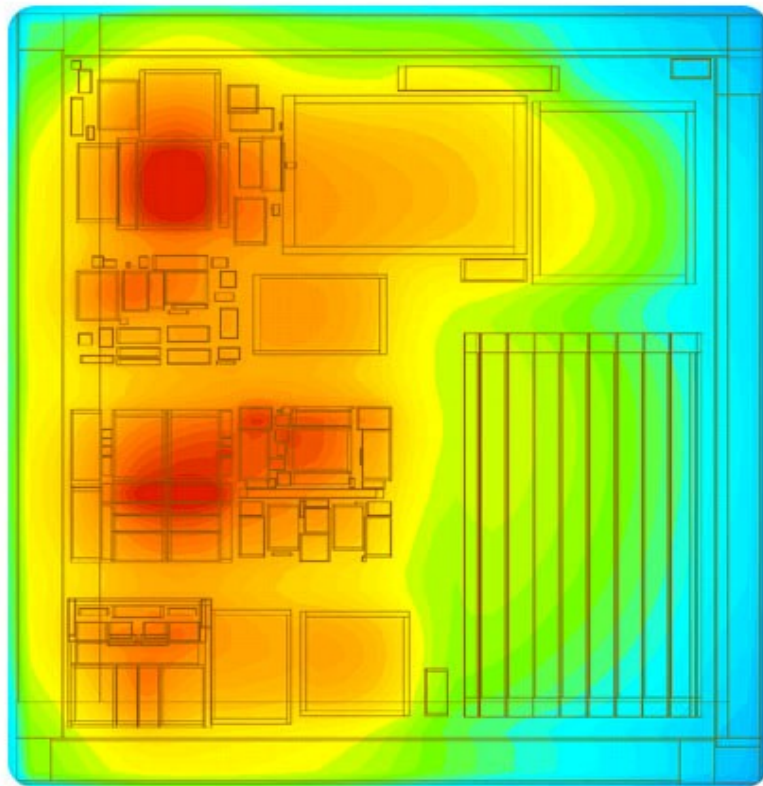


1 of 8
Cores

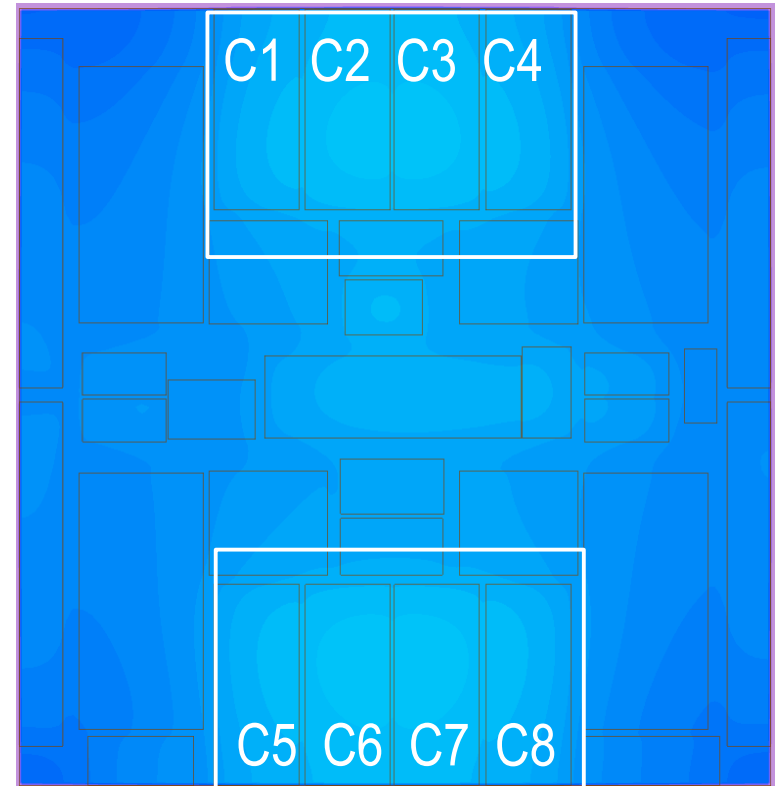


Faster Can Be Cooler

Single-Core Processor



CMT Processor

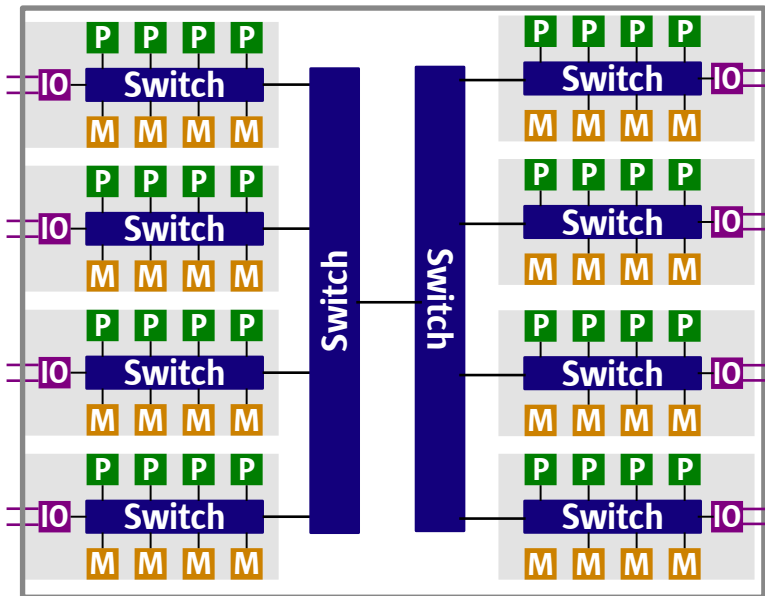


(Not to Scale)

CMT: On-chip = High Bandwidth

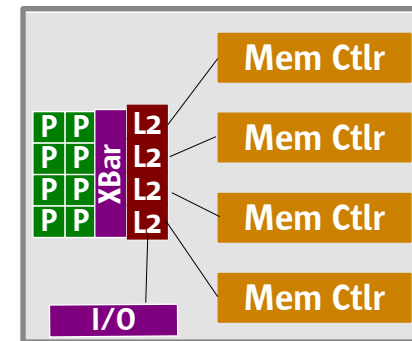
32-thread Traditional SMP System

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

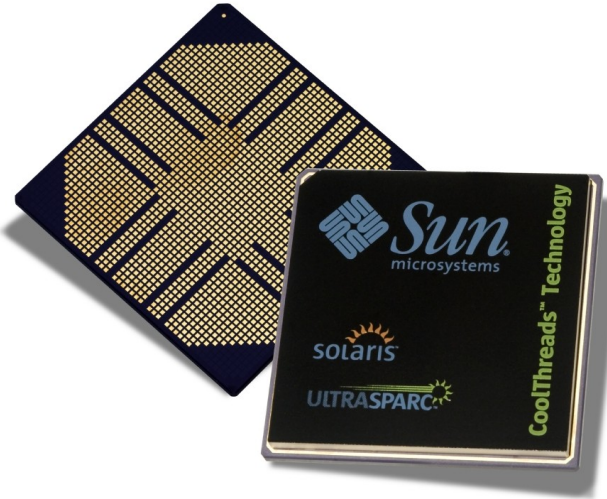
- Fewer servers
- Less floor space
- Reduced power consumption
- Less air conditioning
- Lower administration and maintenance



Reliability

CMT Pays Off with CoolThreads™ Technology

Sun Fire T1000



Sun Fire T2000

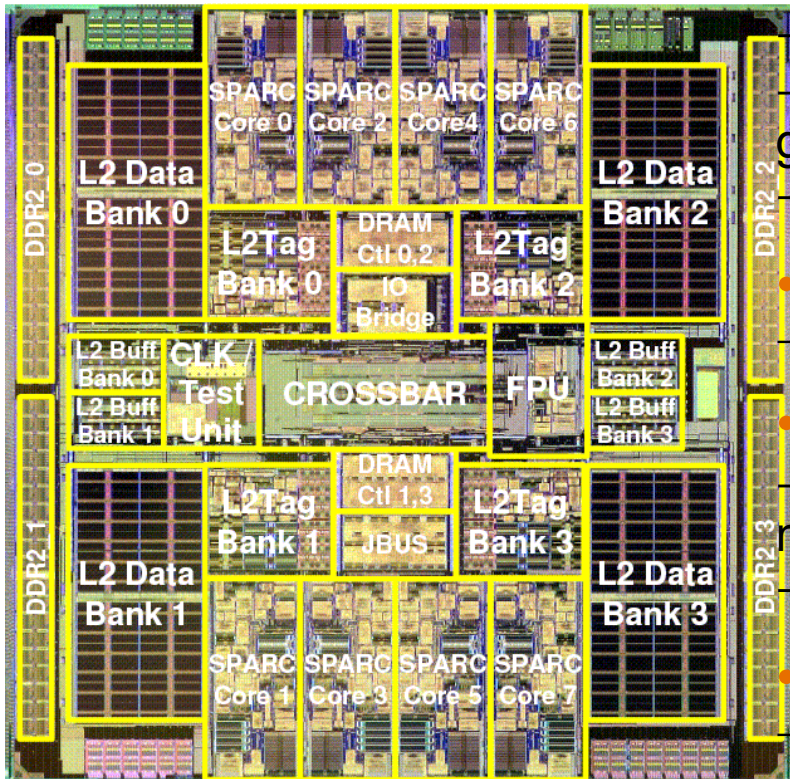


- Up to 5x the performance
- As low as 1/5 the energy
- As small as 1/4 the size



*See disclosures

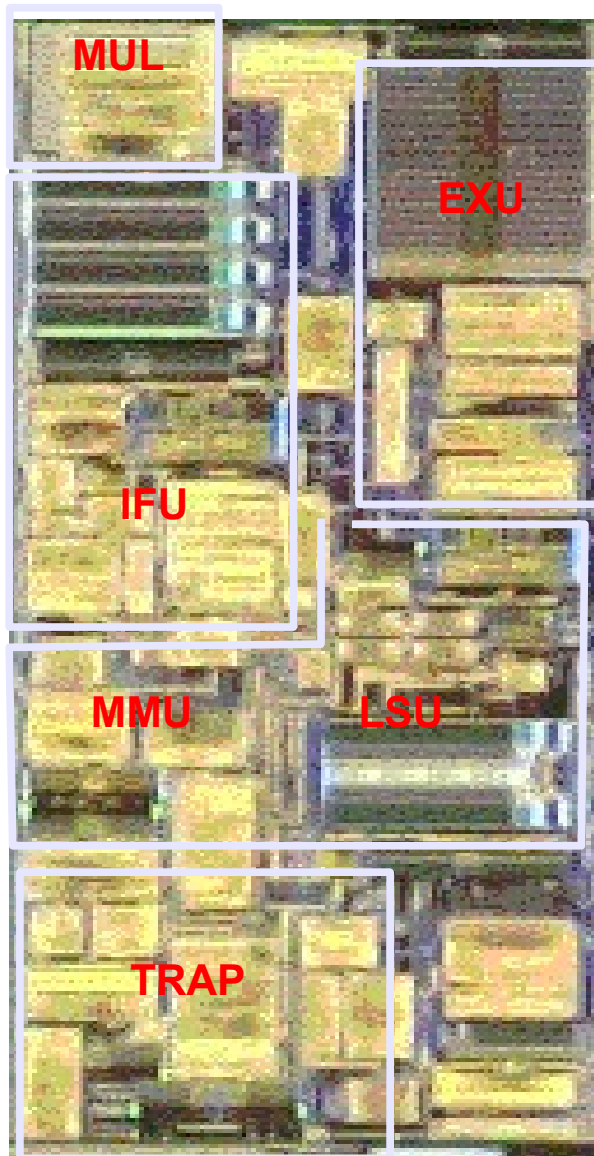
UltraSPARC-T1: Choices & Benefits



- Simple core (6-stage, only 11mm² 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

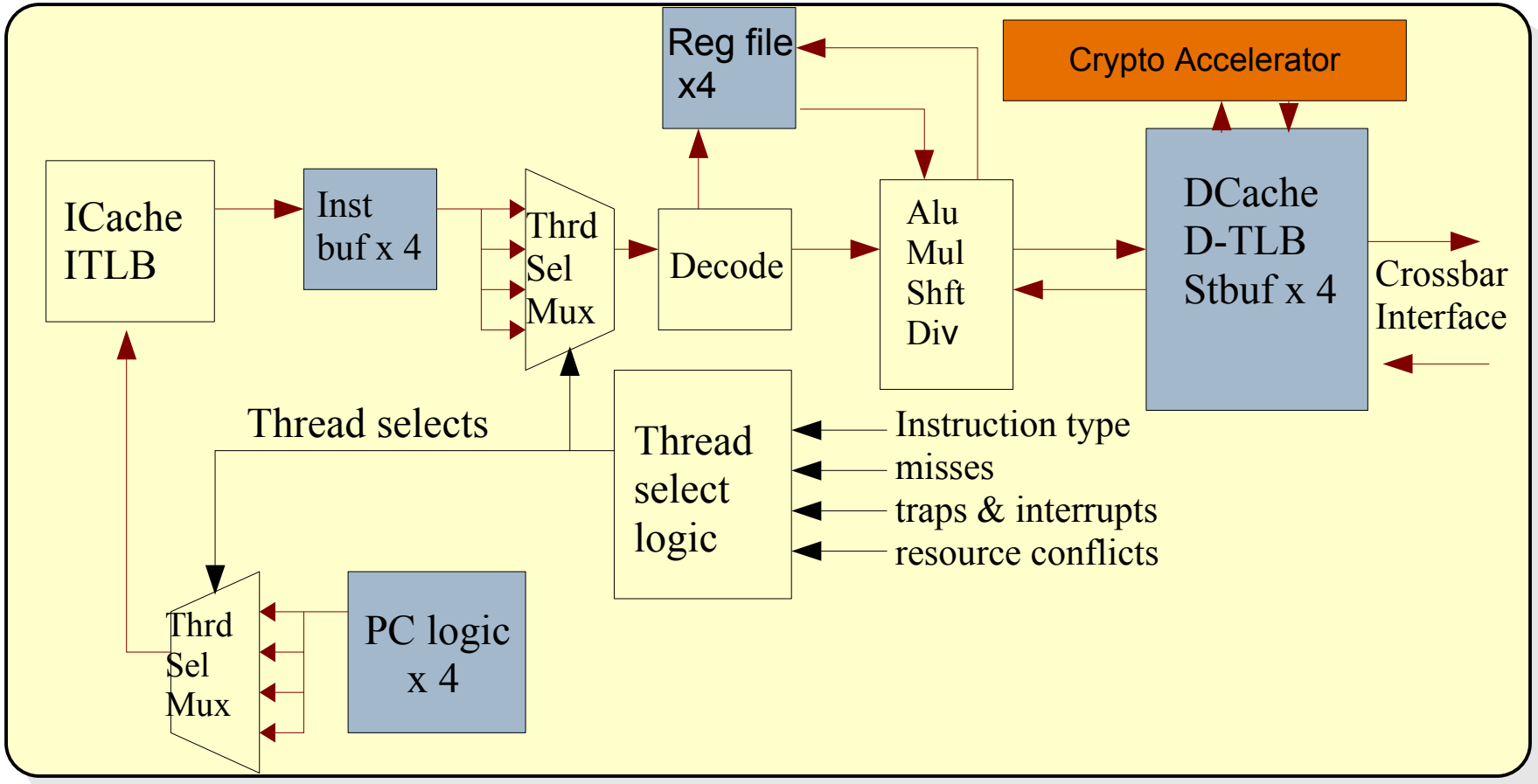
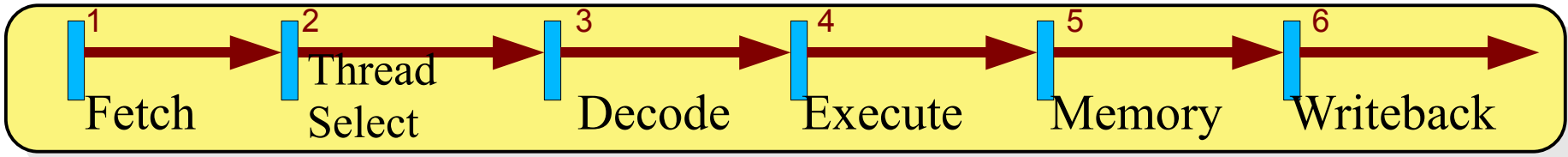
For reference: in 90nm technology, included 8 cores, 32 threads, and only dissipate 70 W

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 = 11mm² in 90nm
- MT adds ~20% area to core

UltraSPARC T1 Processor Core Pipeline



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

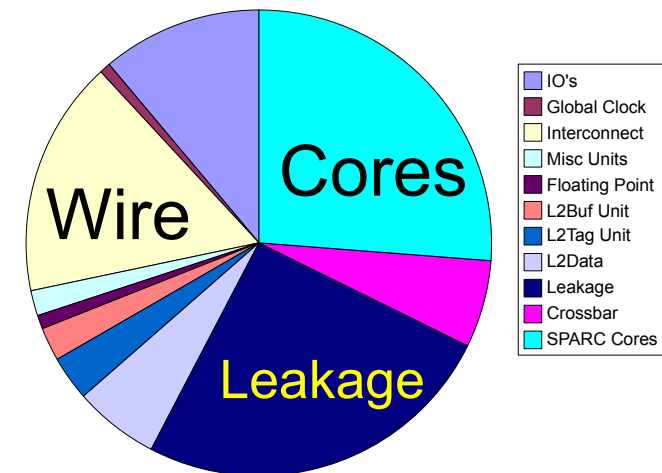
Thread Selection Policy

- 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

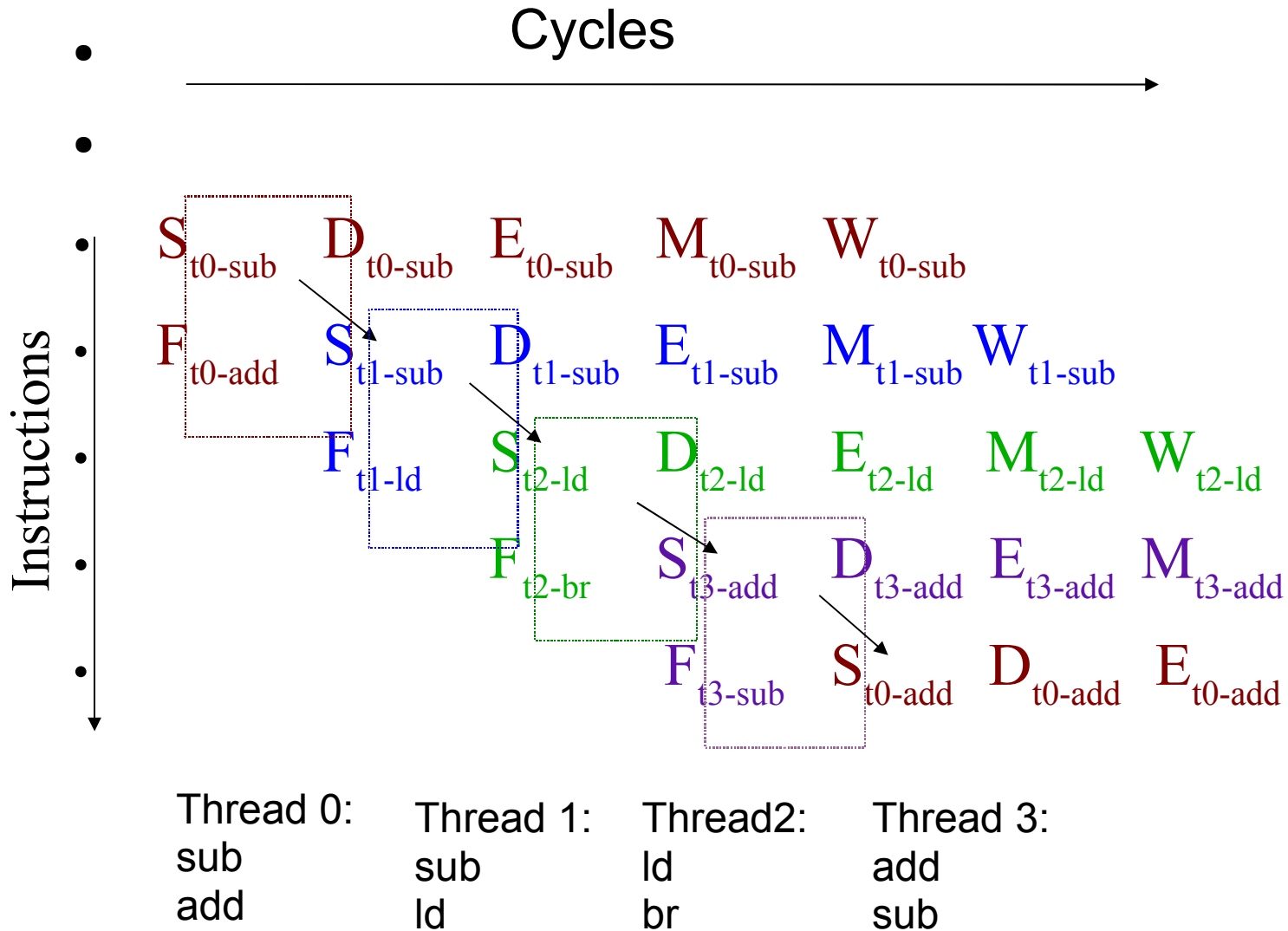
- Power Efficient Architecture
 - Single issue, in-order six stage pipeline
 - Minimal speculation, predication or branch prediction
- Thermal monitoring for power throttling
 - 3 external power throttle pins
- Controlled by thermal diodes
- Stall cycles injected, affecting all threads
 - Memory throttling
- 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

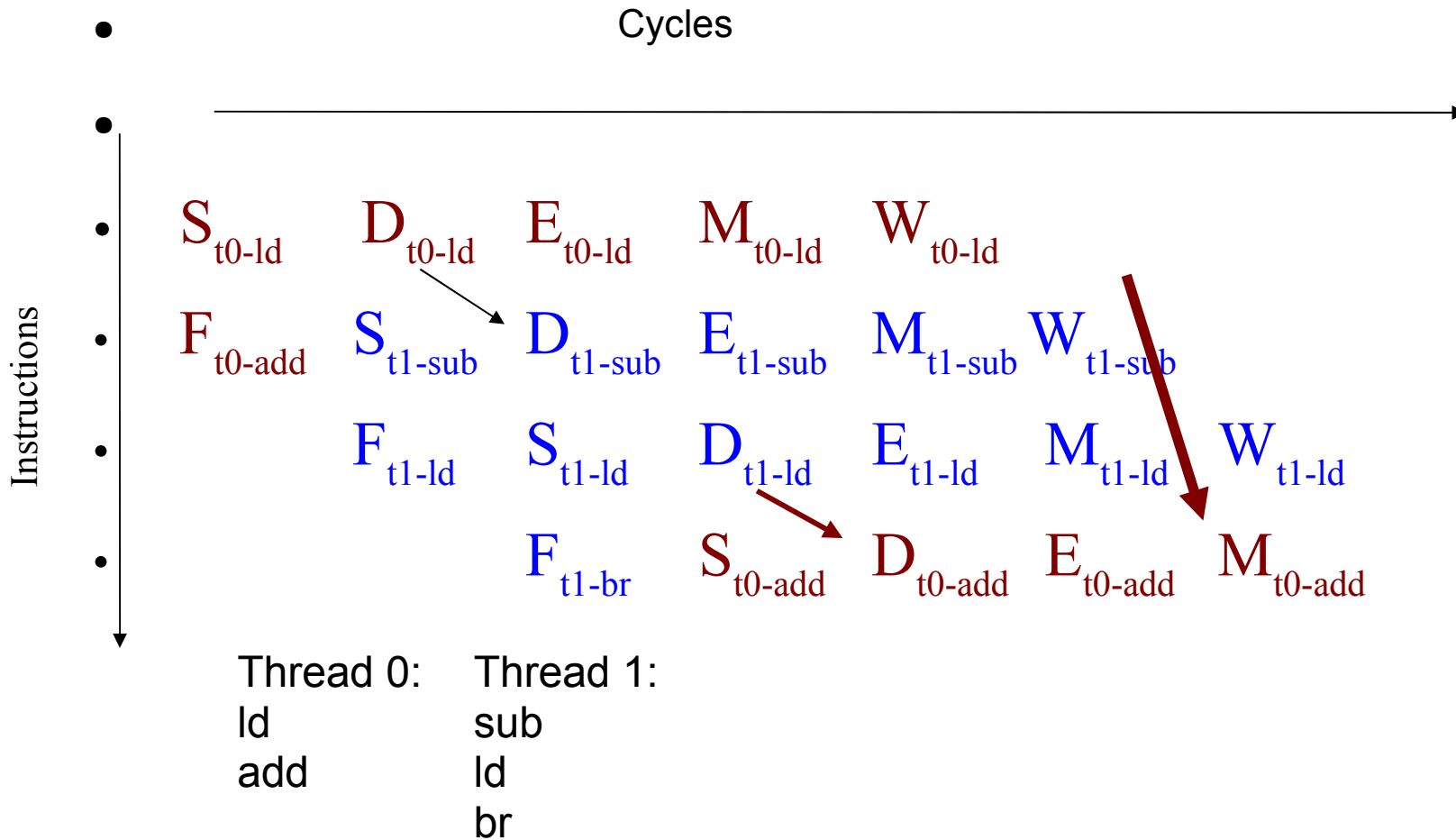


OpenSPARC T1 Microarchitecture

Thread Selection – All Threads Ready



Thread Selection – Two Threads Ready



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

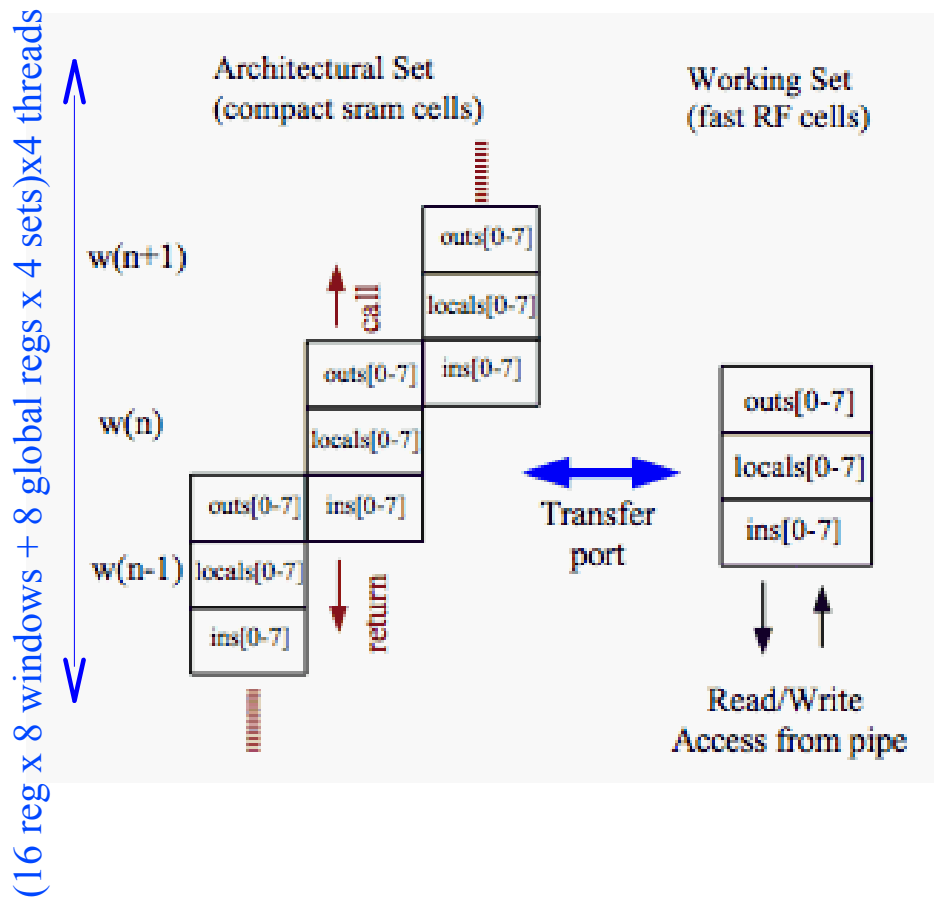
- Fully Associative Instruction TLB

- > 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/clock. 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 $\frac{1}{2}$ 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.

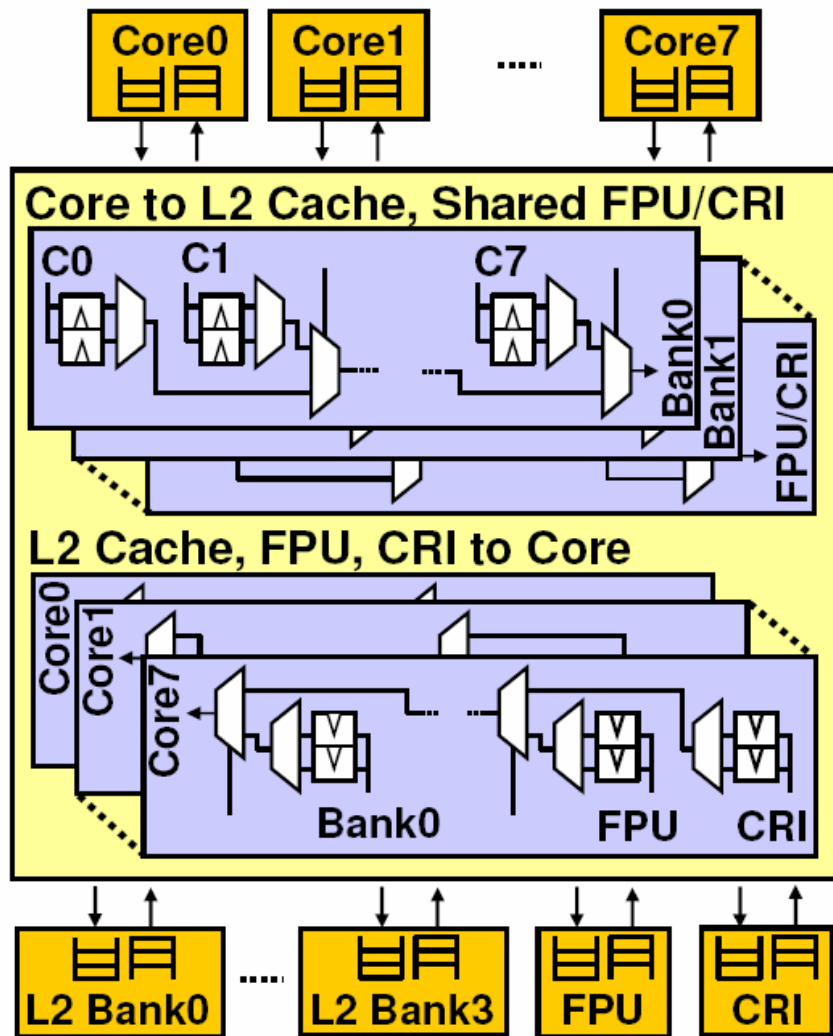
LSU(Cont'd)

- 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,stm,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 $\frac{1}{4}$ 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.

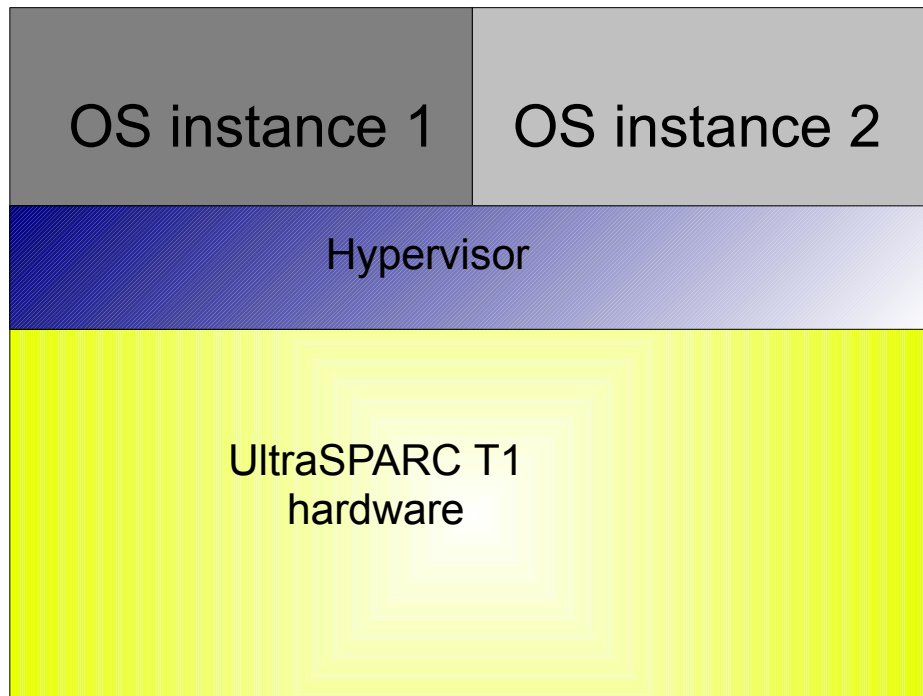
Coherence/Ordering

- 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 T2

T2 Agenda

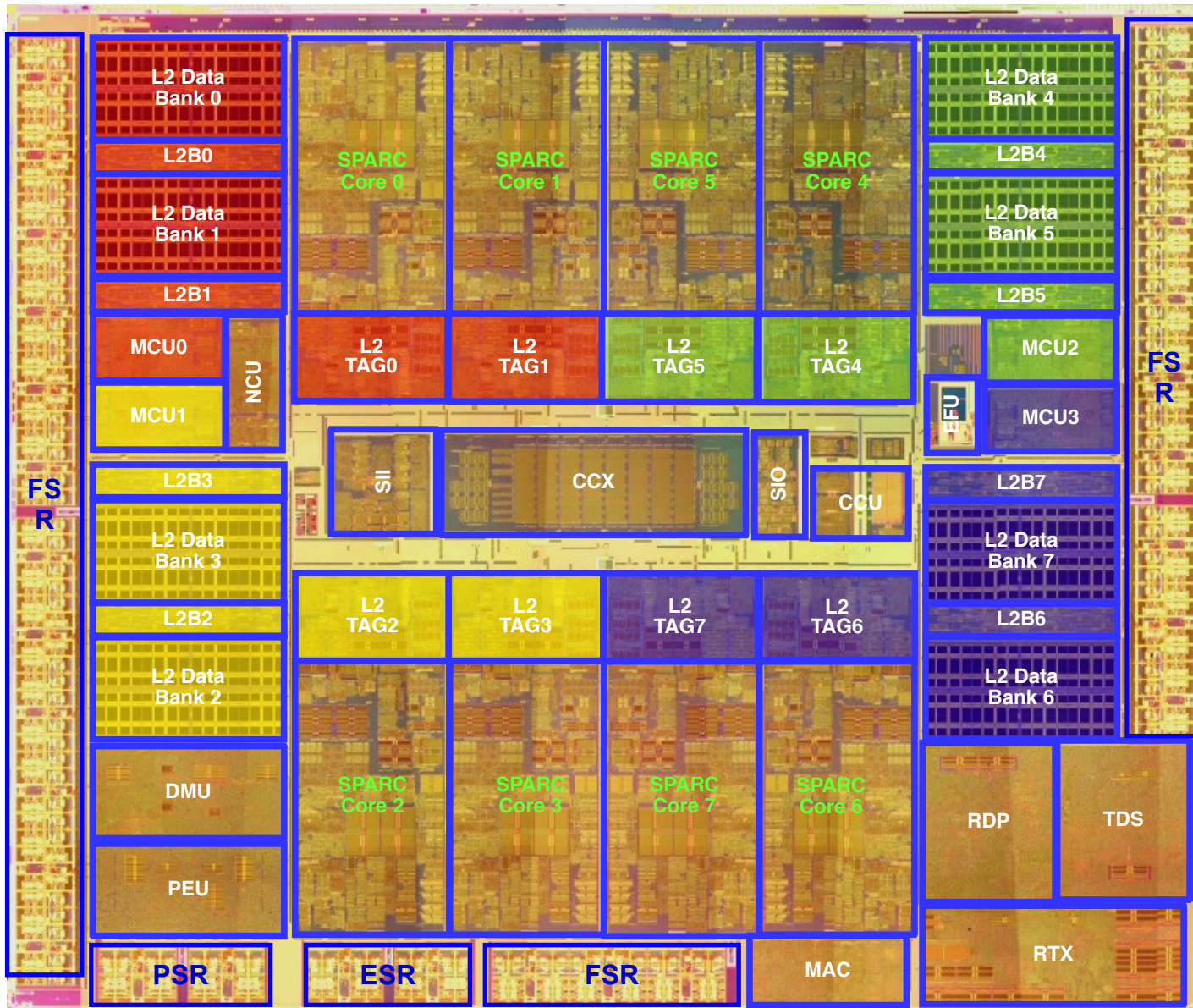
- Chip overview
- SPARC core
 - > Execution Units
 - > Power
 - > RAS
- Crossbar
- L2
- Summary

OpenSPARC T2 Chip Goals

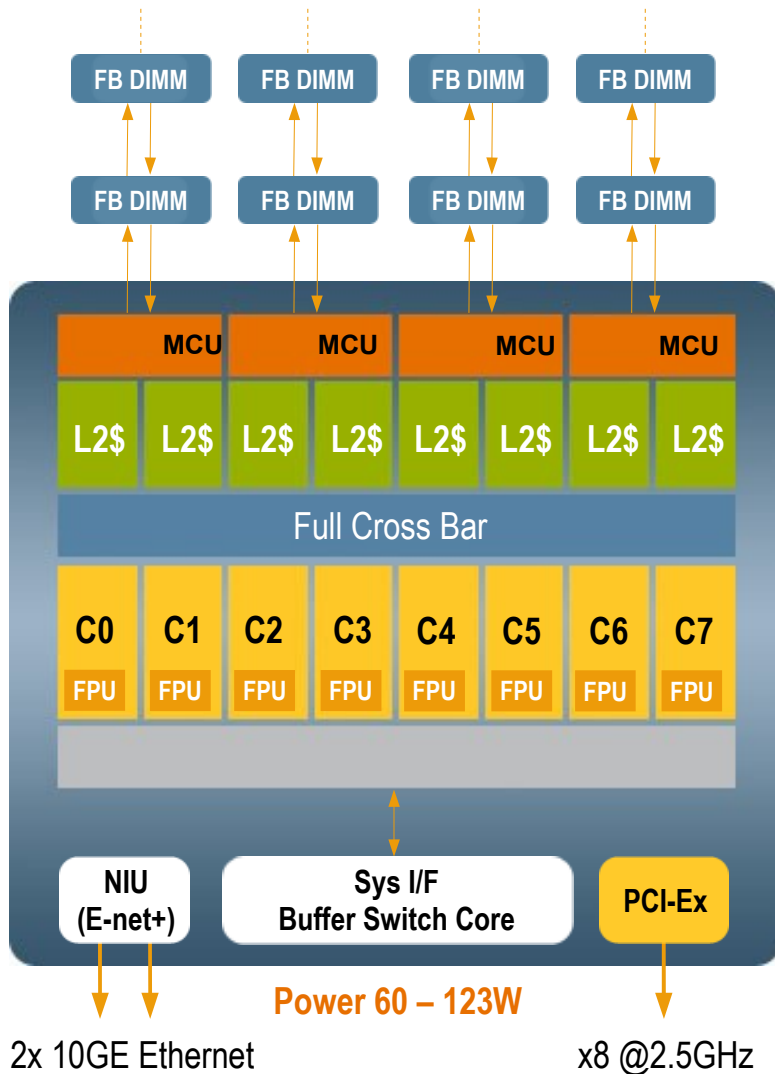
- Double throughput versus OpenSPARC T1
 - > Doubling cores versus increasing threads per core
 - > Utilization of execution units
- Improve throughput / watt
- Improve single-thread performance
- Improve floating-point performance
- Maintain SPARC binary compatibility

UltraSPARC T2 Overview

- 8 SPARC cores,
 - 8 threads each,
 - 64 threads total
- Shared 4MB L2,
 - 8 banks,
 - 16 way associative
- Four dual-channel FBDIMM memory controllers
- Full 8x9 crossbar connects cores to L2 banks / SIU and vice versa
- SIU connects I/O to memory



UltraSPARC[®] T2 Processor: True System On a Chip

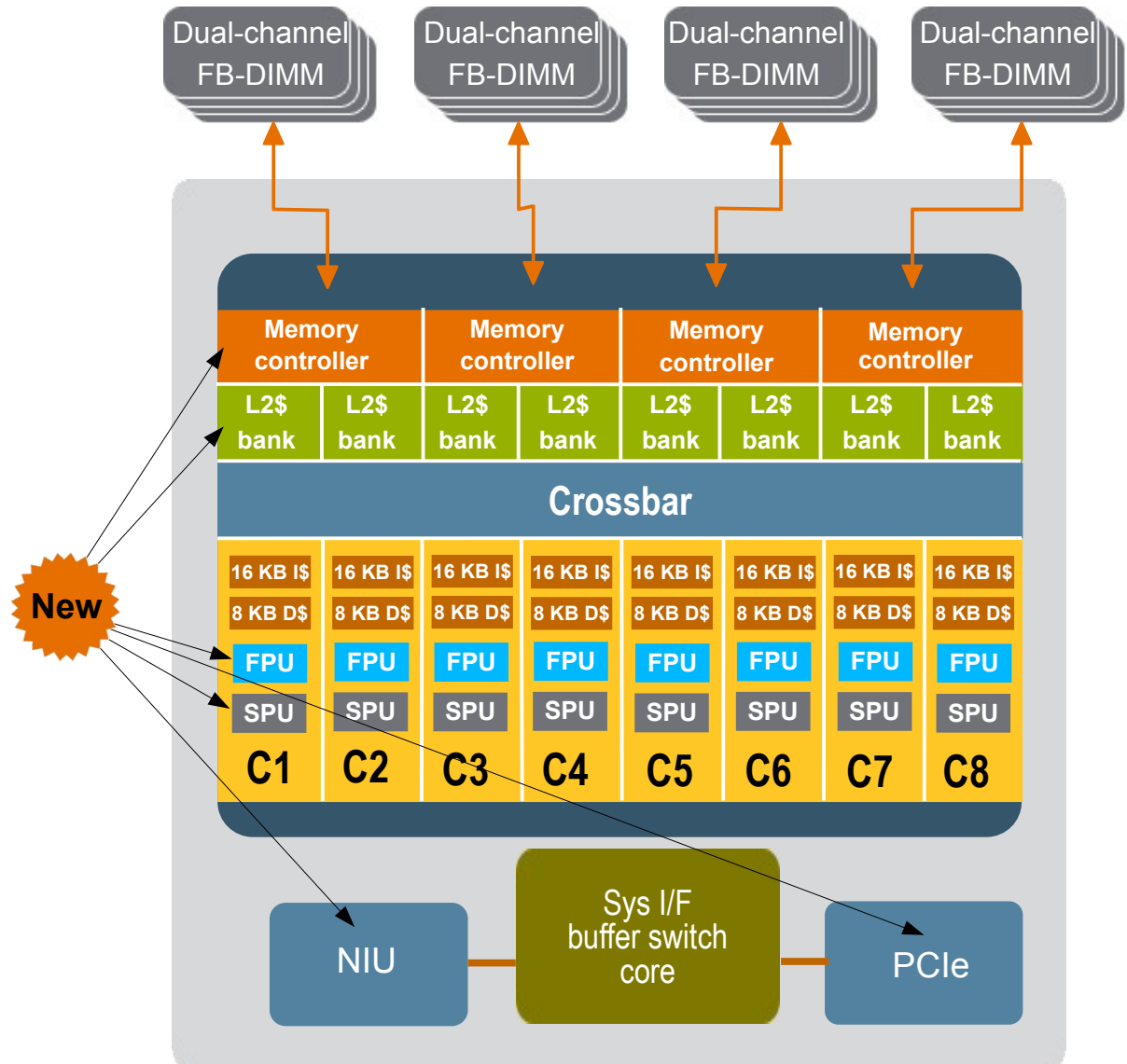


- Up to 8 cores @ 1.2 /1.4GHz
- Up to 64 threads per CPU
- Huge Memory Capacity
 - > Up to 512GB memory
 - > Up to 64 Fully Buffered Dimms
- High Memory Bandwidth
 - > 2.5x memory BW = 60+GB/S
- 8x FPUs, 1 fully pipelined floating point unit/core
- 4MB L2\$ (8 banks) 16 way
- Security co-processor / core
 - > DES, 3DES, AES, RC4, SHA1, SHA256, MD5, RSA to 2048 key, ECC, CRC32

UltraSPARC T2 Architecture

A true system on a chip

- Up to 8 SPARC cores @ 1.0–1.4 GHz
 - > Up to 64 total threads
 - > 4-MB, 16-way, 8-bank L2\$
- 1 floating-point unit per core
- 1 SPU (crypto) per core
- FB-DIMM 1.0 support
- 8-lane PCI Express 1.0 bus interface
- 2 x 1/10 Gb on-chip Ethernet
- Power: < 95 W (nominal)



UltraSPARC T2 “Zero Cost” Security

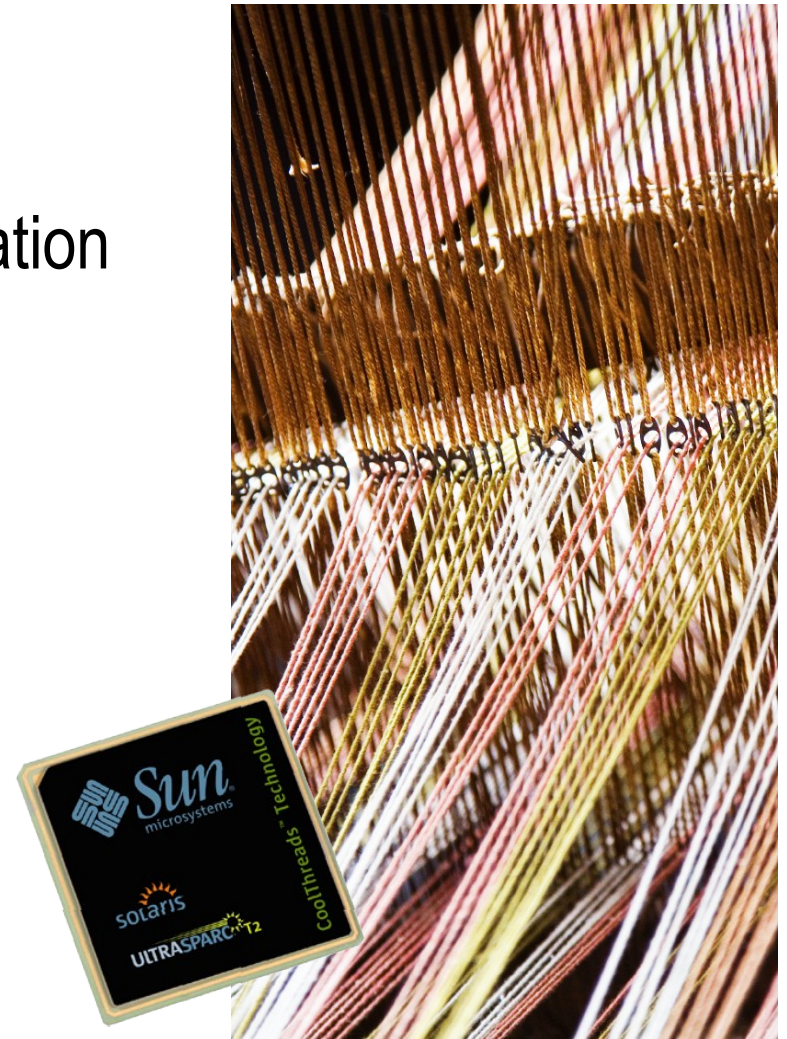


- One crypto unit integrated per core (eight total)
- Supports the ten most common ciphers and secure hashing functions
- Composed of two independent sub-units that operate in parallel
 - > Modular Arithmetic Unit
 - > Cipher/Hash Unit



Integrated Multithreaded 10 GbE

- Dual, multithreaded, 10 GbE (XAUI)
 - > Up to 4X the performance of current network interface cards
 - > 16 Rx and Tx DMA channels for virtualization
- Limited classification
 - > Classified at layer 2 ,3 and 4 into Rx DMA buffer to match the flow
- Benefits
 - > Eliminates network I/O bottlenecks
 - > Enables faster network access



Integrated Floating Point Unit

- Each UltraSPARC T2 core has its own Floating Point Unit
- Fully-pipelined (except divide/sqrt)
 - > Divide/sqrt in parallel with add or multiply operations of other threads
- Full VIS 2.0 implementation
- FPU performs integer multiply, divide, population count



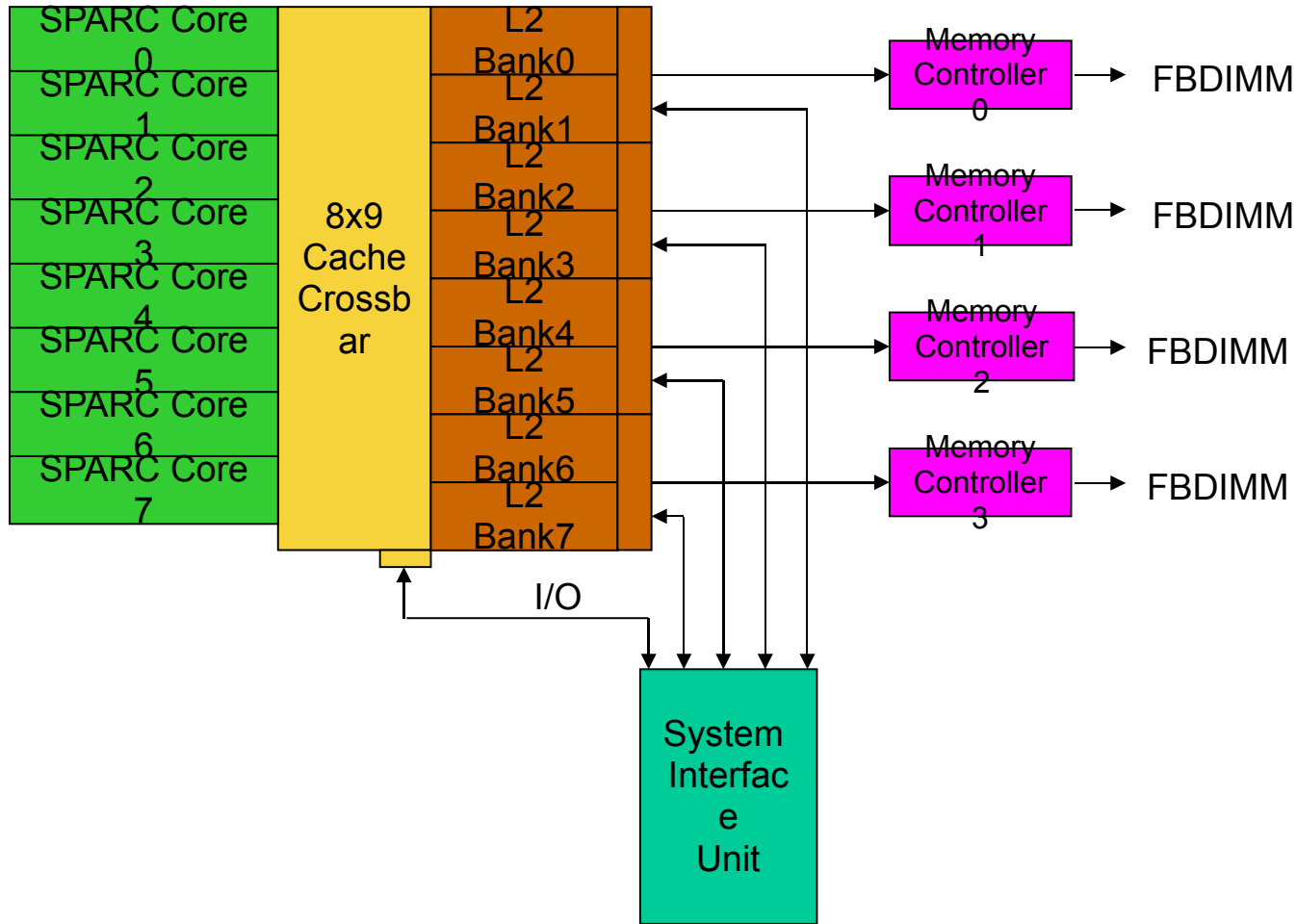
UltraSPARC T2: 7 World Records

Built on a heritage of network throughput

- Standard performance benchmarks
 - > SPECint_Rate2006 (single chip)
 - > SPECfp_Rate2006 (single chip)
 - > Web Performance: SPECweb2005
 - > Unix Java VM (single socket): SPECjbb2005
 - > Java App Server: SPECjAppServer2004 (dual node)
 - > Unix ERP Platform: Single-socket
SAP SD-2 Tier
 - > OLTP Platform: Database Tier
SPECjAppServer2004 Dual Node Result



OpenSPARC T2 Block Diagram



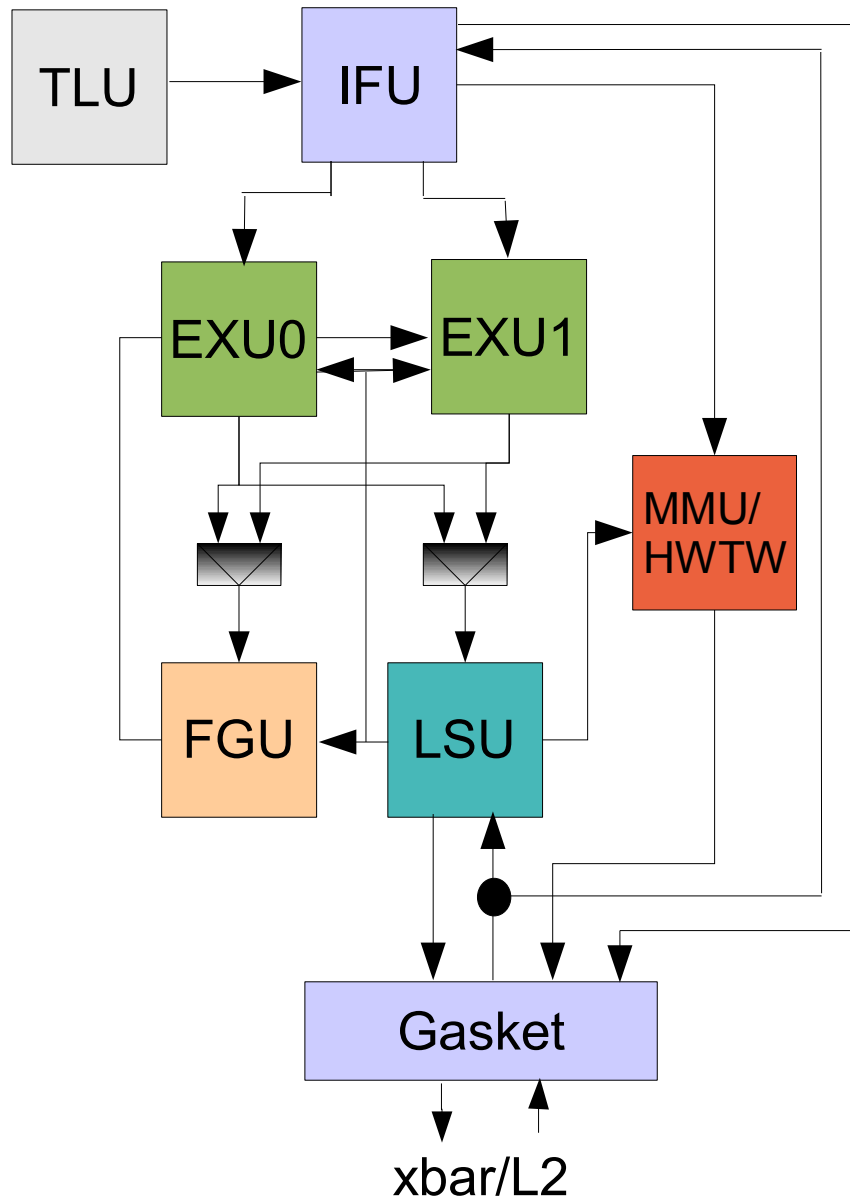
OpenSPARC T1 to T2 Core Changes

- Increase threads from 4 to 8 in each core
- Increase execution units from 1 to 2 in each core
- Floating-point and Graphics Unit in each core
- New pipe stage: pick
 - > Choose 2 threads out of 8 to execute each cycle
- Instruction buffers after L1 instruction cache for each thread
- Increase set associativity of L1 instruction cache to 8
- Increase size of fully associative DTLB from 64 to 128 entries
- Hardware tablewalk for ITLB and DTLB misses
- Speculate branches not taken

OpenSPARC T1 to T2 Chip Changes

- Increase L2 banks from 4 to 8
 - > 15 percent performance loss with only 4 banks and 64 threads
- FBDIMM memory interface replaces DDR2
 - > Saves pins
 - > Improved bandwidth
 - > 42 GB/sec read
 - > 21 GB/sec write
 - > Improved capacity (512 GB)
- RAS changes (to match T1 FIT rate)

SPARC Core Block Diagram



- IFU – Instruction Fetch Unit
 - > 16 KB I\$, 32B lines, 8-way SA
 - > 64-entry fully-associative ITLB
- EXU0/1 – Integer Execution Units
 - > 4 threads share each unit
 - > Executes one instruction/cycle
- LSU – Load/Store Unit
 - > 8KB D\$, 16B lines, 4-way SA
 - > 128-entry fully-associative DTLB
- FGU – Floating-Point and Graphics Unit
- TLU – Trap Logic Unit
 - > Updates machine state, handles exceptions and interrupts
- MMU – Memory Management Unit
 - > Hardware tablewalk (HWTW)
 - > 8KB, 64KB, 4MB, 256MB pages
- Gasket arbitrates between the core units for the crossbar interface

SPARC Core Pipeline

- 8 stage integer pipeline



- > 3 cycle load-use penalty

- > Memory (data address translation, access tag/data array)

- > Bypass (late way select, data formatting, data forwarding)

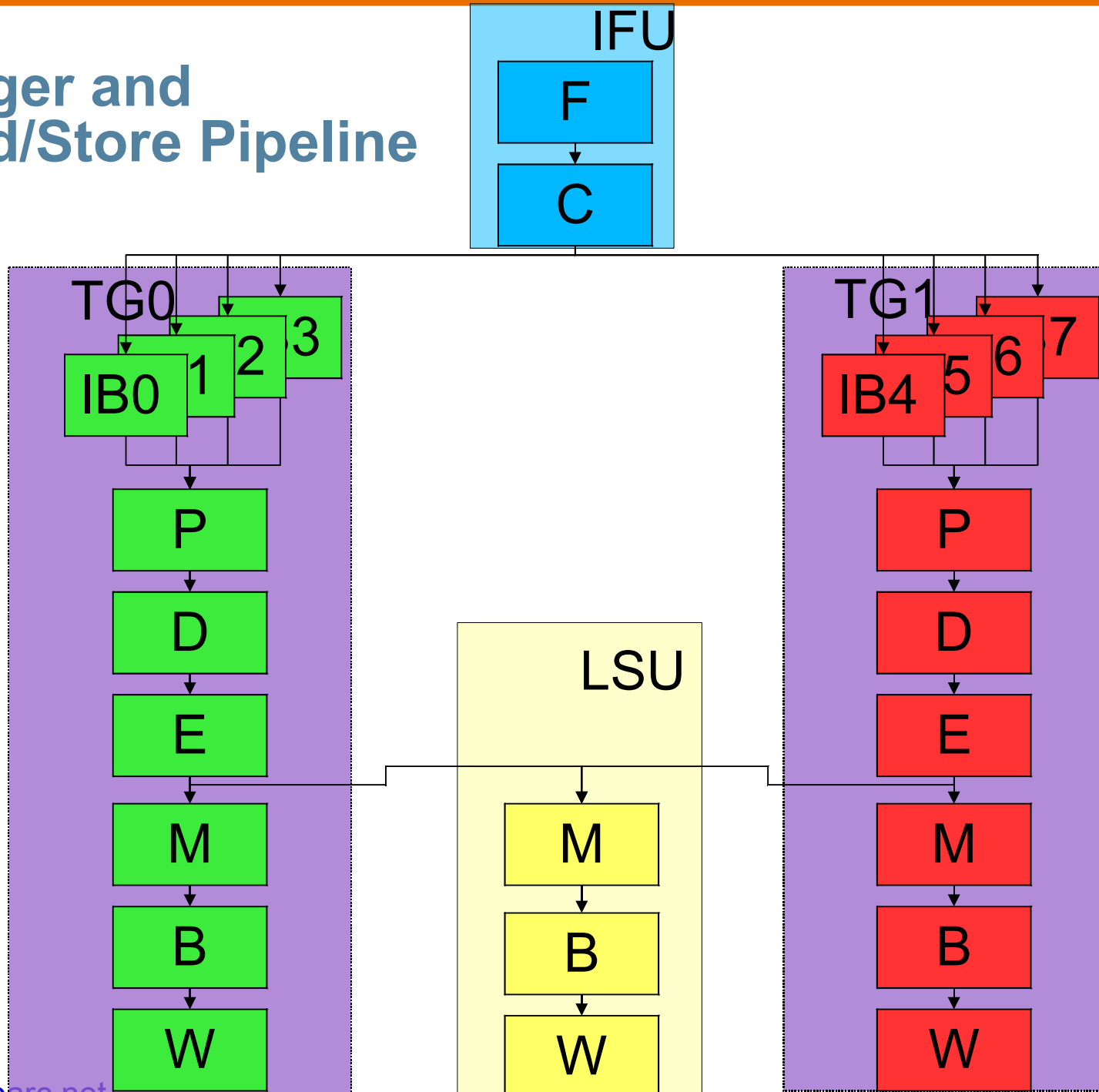
- 12 stage floating-point pipeline



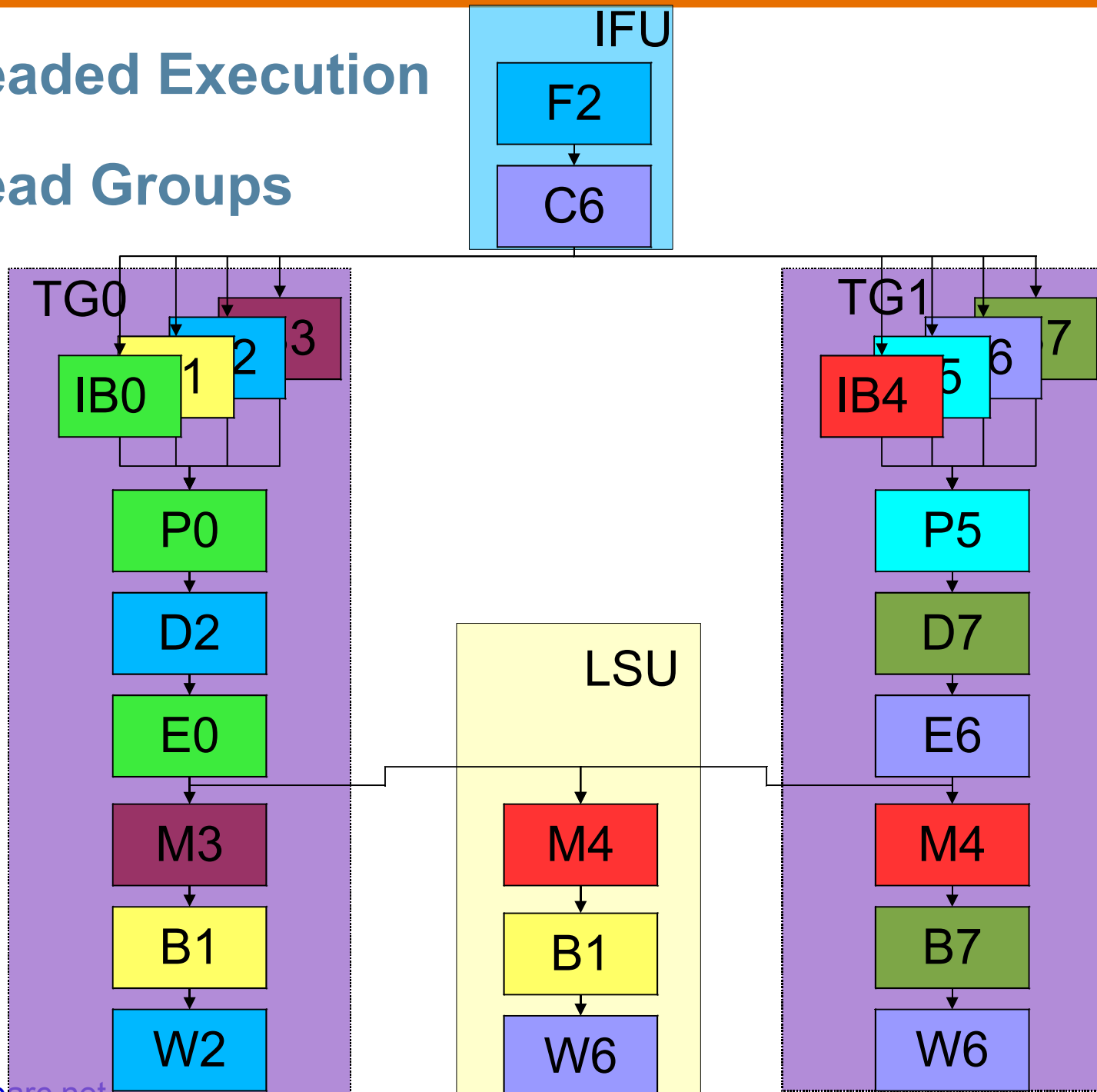
- > 6 cycle latency for dependent FP instructions

- > Longer pipeline for divide/sqrt

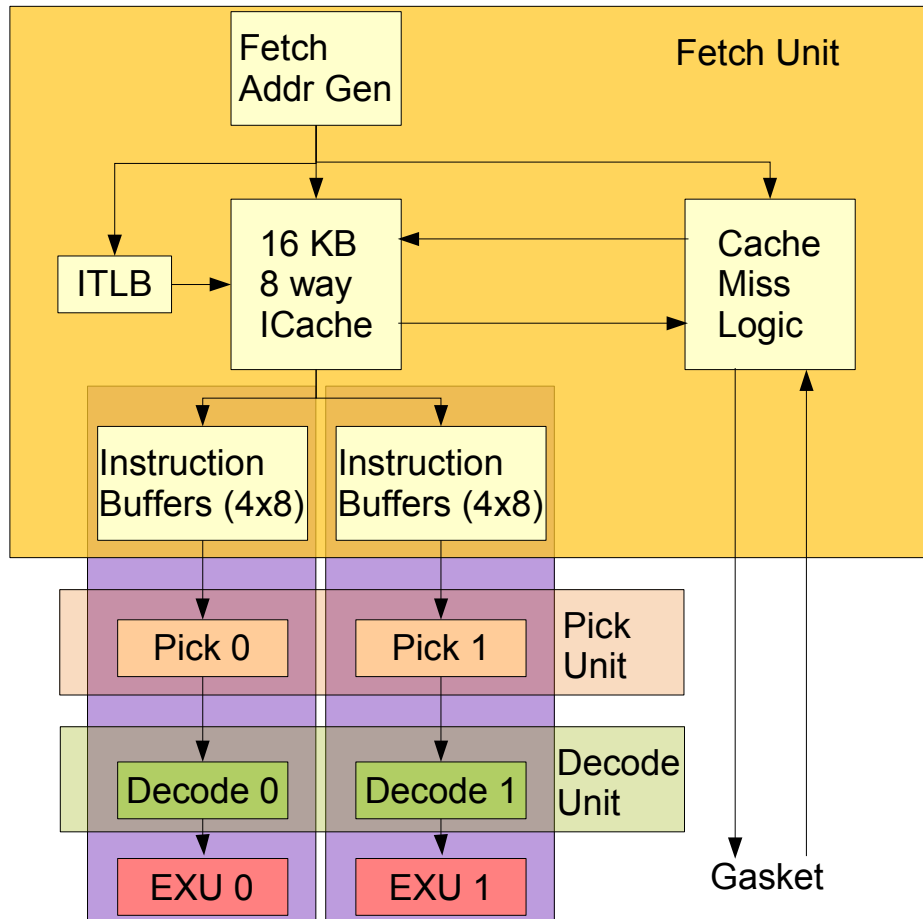
Integer and Load/Store Pipeline



Threaded Execution and Thread Groups

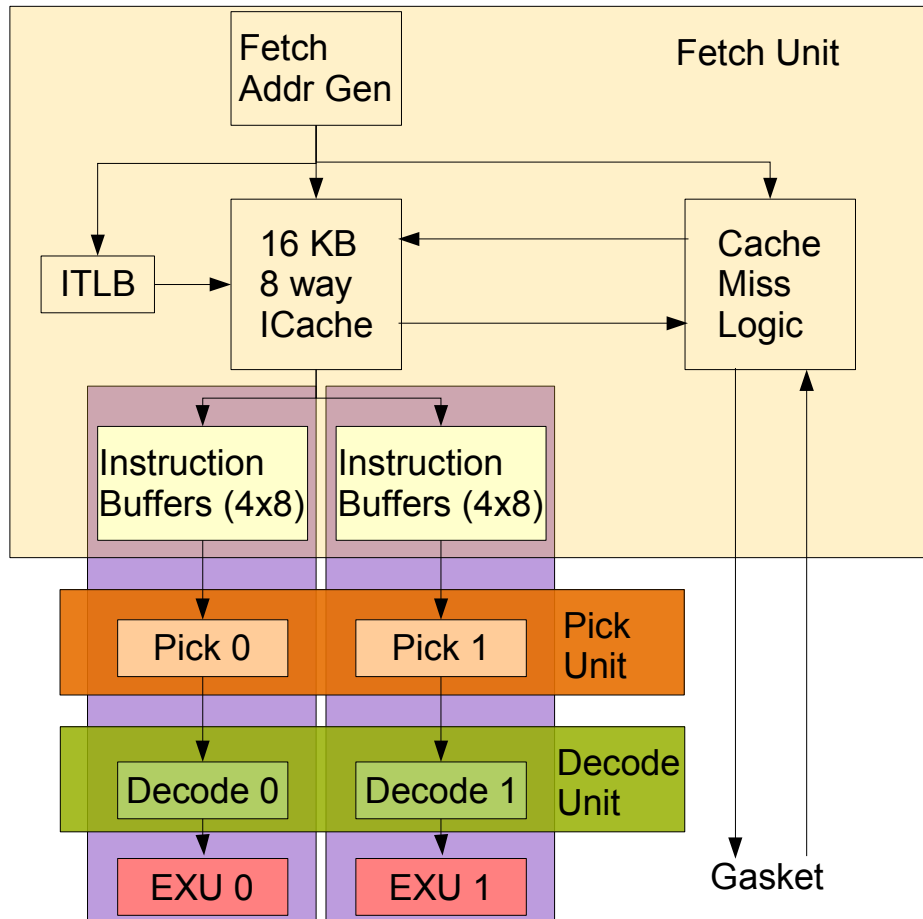


Instruction Fetch



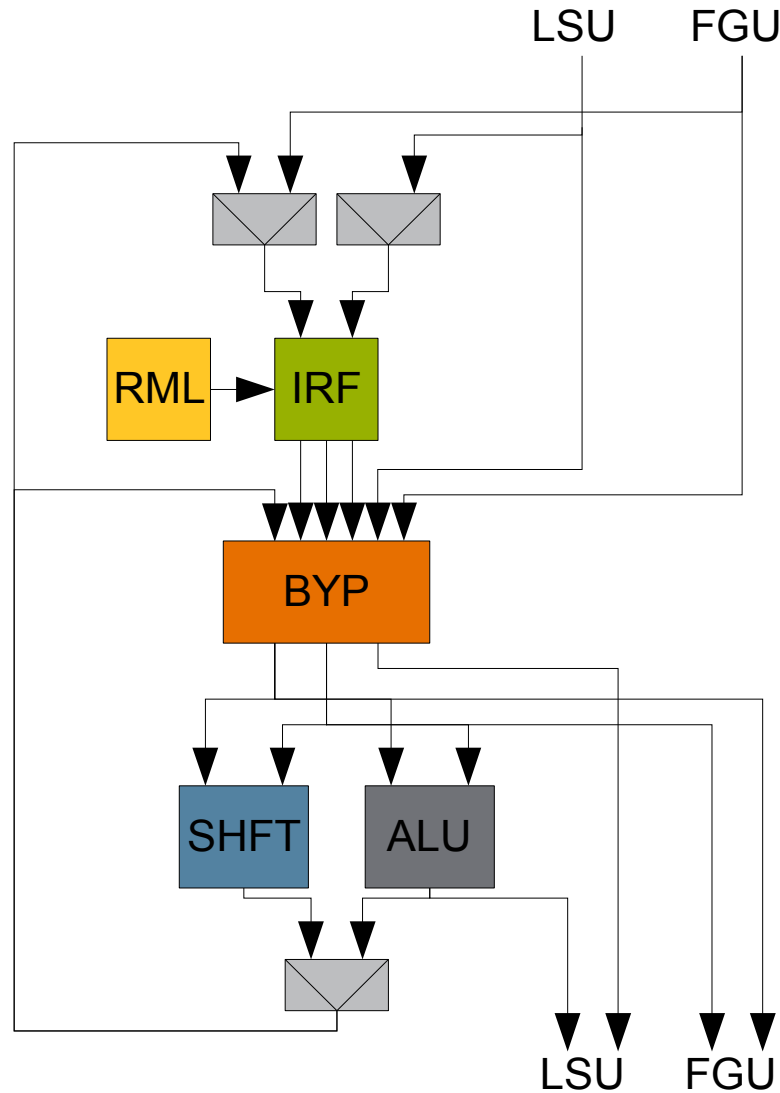
- Instruction cache and fetch shared between the eight threads
- Fetch up to four instructions per cycle
 - > Each thread in ready or wait state
 - > Wait state caused by:
 - > TLB miss
 - > cache miss
 - > instruction buffer full
 - > Least-recently fetched among ready threads
 - > One instruction buffer/thread
- Branches assumed to be not-taken;
 - 5-cycle penalty if taken
- > T1 switched threads if branch or load fetched
- Limited I\$ miss prefetching
- Pick and Decode decoupled from Fetch by the instruction buffer

Instruction Pick and Decode



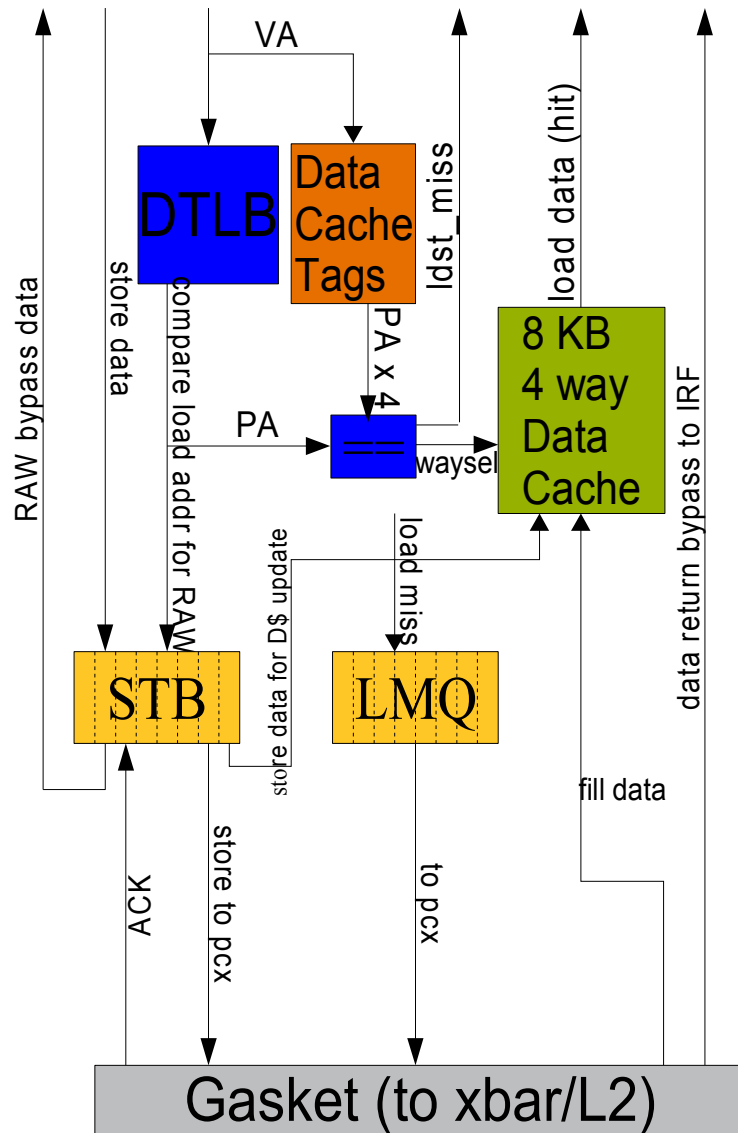
- Threads divided into two groups of four threads each
- One instruction from each thread group picked each cycle
 - > Least-recently picked within a thread group among ready threads
 - > Wait states: dependency, D\$ miss, DTLB miss, divide/sqrt, ...
 - > Gives priority to nonspeculative threads (e.g. no load)
- Decode resolves conflicts
 - > Each thread group picks independently of the other
 - > Both thread groups pick load/store or FGU instructions
- Independent instructions after loads

Execution Unit



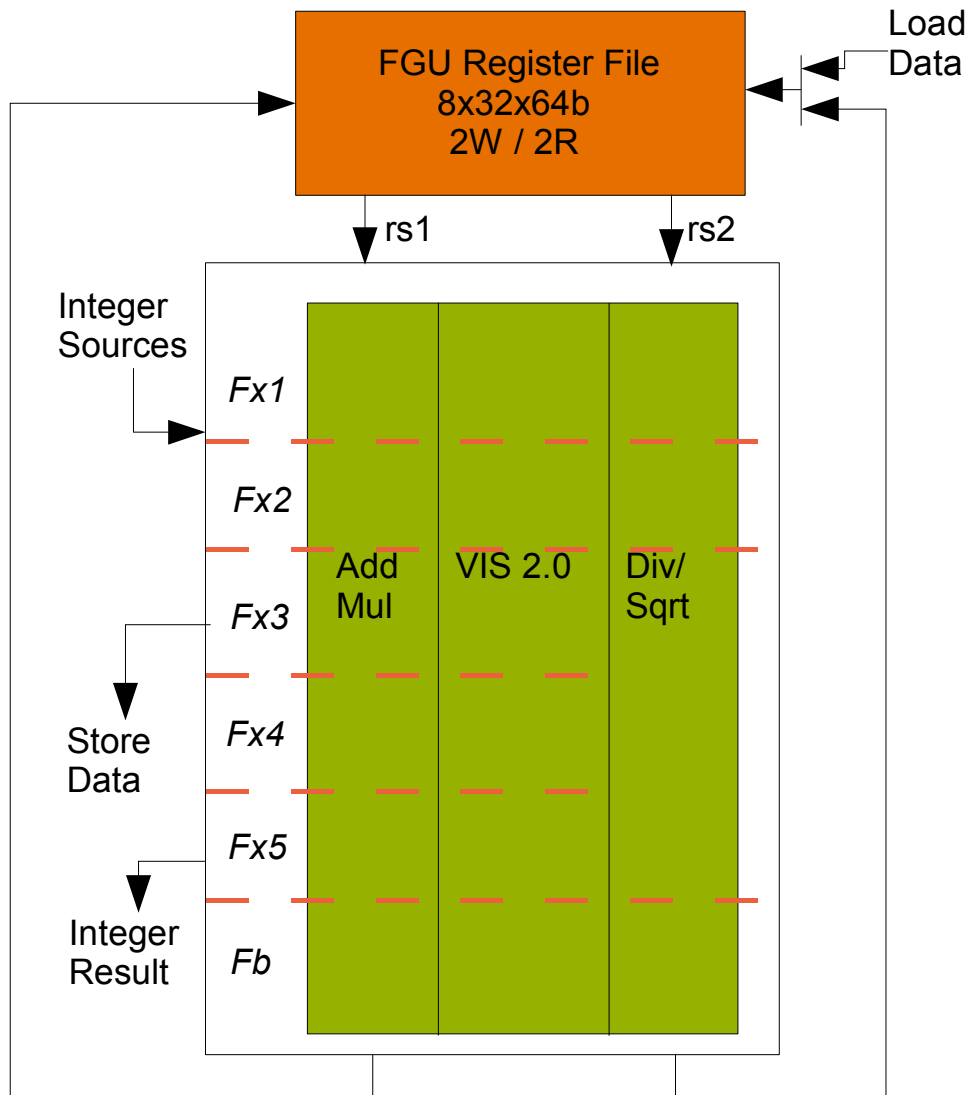
- Executes integer operations and some graphics operations
- Generates addresses for loads and stores
- Adder / logic unit, shifter
- Each EXU contains state for
 - four threads
- > Integer register file (IRF)
- > 8 register windows per thread
- > 4 global levels per thread
- > Window or global level change requires multiple cycles
 - (but pipelined)
- > Register window management logic (RML)

Load Store Unit



- One load or store per cycle
- Store-through
- D\$ allocates on load misses, updates on store hits
- Load Miss Queue (LMQ) supports one pending load miss per thread
- Store buffer (STB) contains
 - 8 stores per thread
- > Stores to same L2 cache line are pipelined to L2
- Arbiter for crossbar between load misses and stores
- > Fairness between threads, loads, and stores

Floating-point and Graphics Unit



- Fully pipelined
 - (except divide/sqrt)
- > Divide/sqrt in parallel with add or multiply operations of other threads
- FGU performs integer multiply, divide, population count
- FGU predicts exceptions in Fx1 stage

Memory Management Unit

- Hardware tablewalk of up to 4 translation storage buffers (TSBs) (a.k.a page tables)
 - > Each TSB supports one page size
- Three search modes:
 - > Sequential – search TSBs in order
 - > Burst – search TSBs in parallel
 - > Prediction – use VA to predict TSB to search
 - > Two-bit predictor orders first two TSB searches
- Up to 8 pending misses
 - > ITLB or DTLB miss per thread

Core Power Management

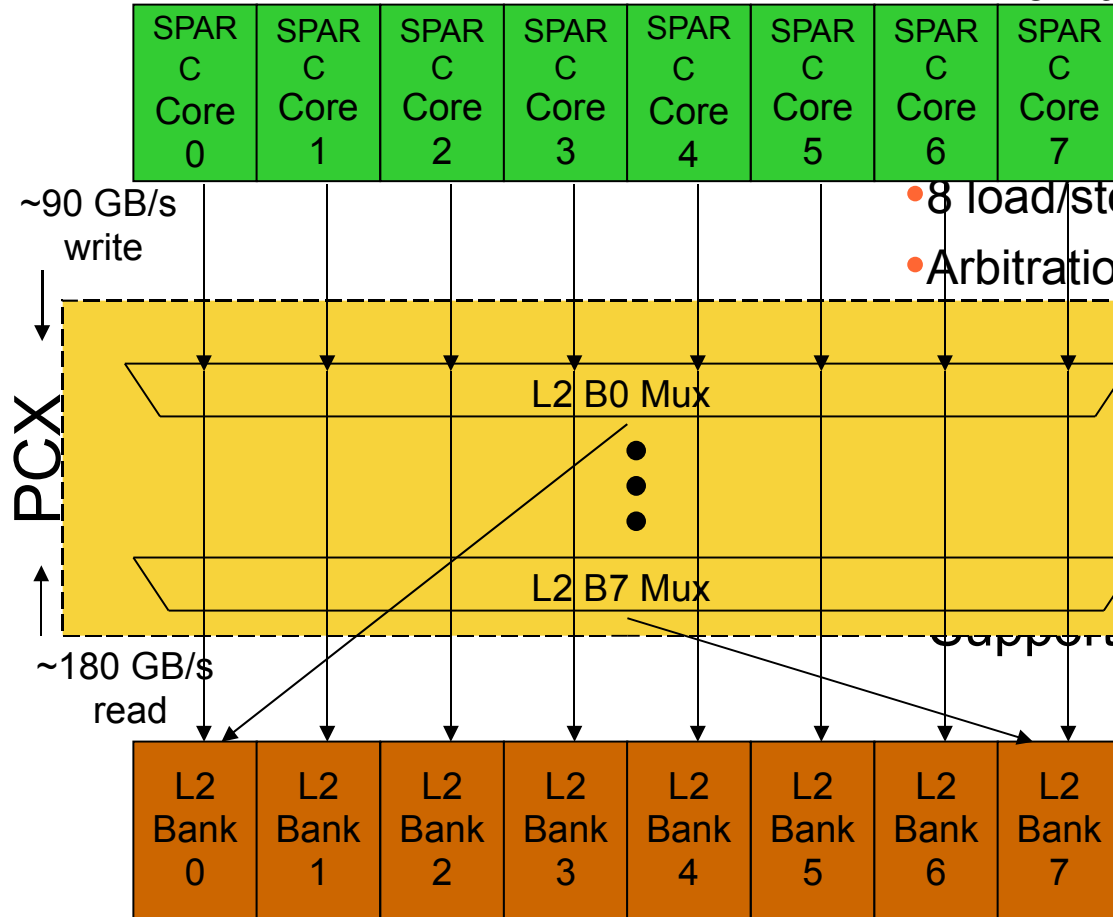
- Minimal speculation
 - > Next sequential I\$ line prefetch
 - > Predict branches not-taken
 - > Predict loads hit in D\$
 - > Pick independent instructions after loads
 - > Hardware tablewalk search control
- Extensive clock gating
 - > Datapath
 - > Control blocks
 - > Arrays
- External power throttling
 - > Add stall cycles at decode stage

Core Reliability and Serviceability

- Extensive RAS features
 - > Parity-protection on I\$, D\$ tags and data, ITLB, DTLB CAM and data, store buffer address
 - > ECC on integer RF, floating-point RF, store buffer data, trap stack, other internal arrays
- Combination of hardware and software correction flows
 - > Hardware re-fetch for I\$, D\$
 - > ECC inside the core is corrected by software

Crossbar

- Two complementary,
 - non-blocking, pipelined switches



processor to cache

cache to processor

- 8 load/store requests and 8 data returns can be done at

- Arbitration for a target is required

to oldest requestor to maintain fairness and

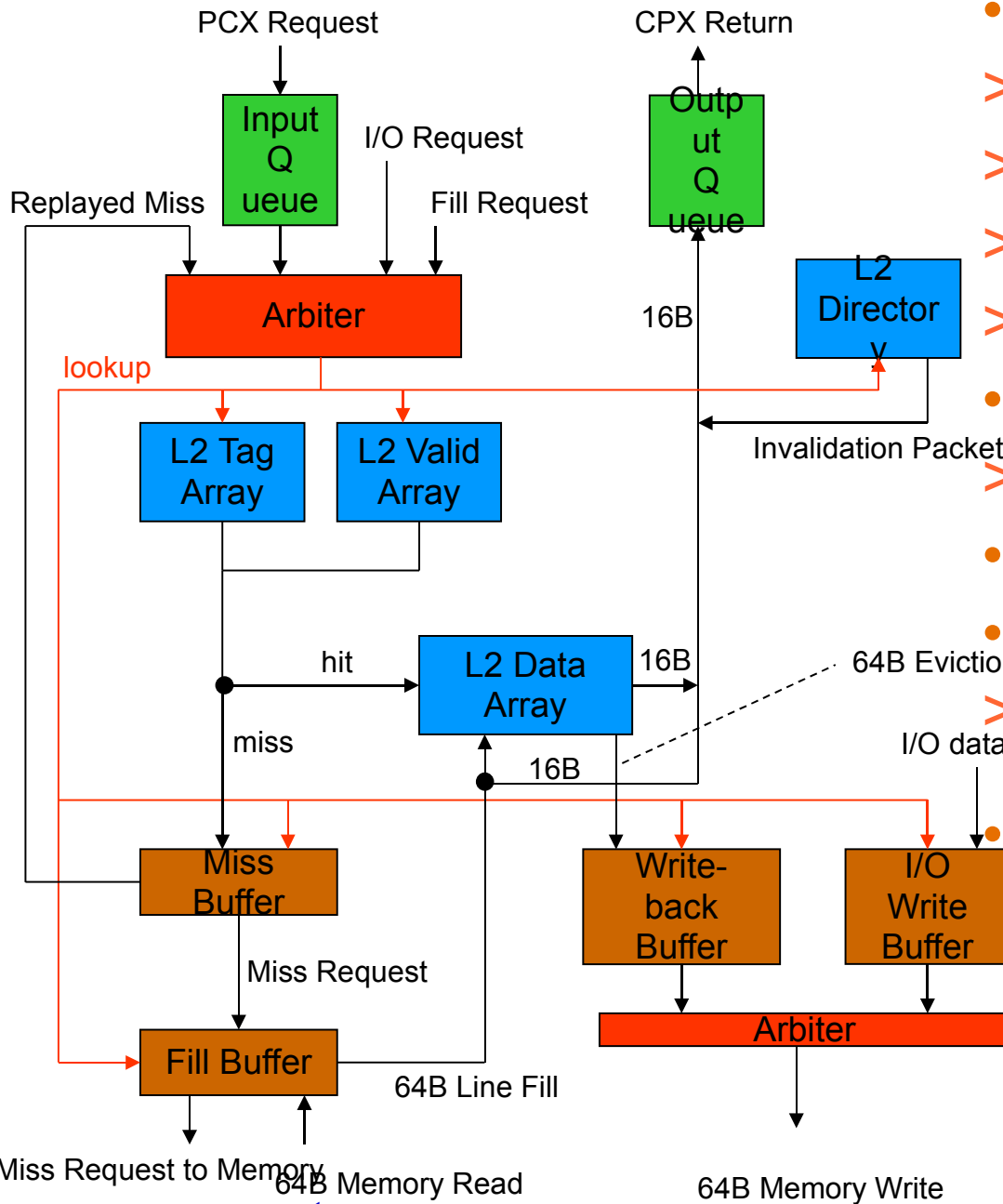
arbitration protocol

bitrate, and grant

byte writes from a core to a bank

supports 16 byte reads from a bank to core

L2 Cache



- 4 MB L2 cache
- > 16 way set associative
- > 8 L2 banks
- > 64 byte line size
- > T1: 3 MB, 12 ways, 4 banks
- L2 cache is write-back, write-allocate
- > L1 data cache is write-thru
- Support for partial stores
- L2 cache manages coherency
- > Maintains directories for all
- 16 L1 caches
- 16 byte data transfers to the cores

write-allocate

Miss Request to Memory 64B Memory Read

64B Memory Write

Summary

- >2x throughput and throughput/watt vs. OpenSPARC T1
- Greatly improved floating-point performance
- Significantly improved integer performance



OpenSPARC™

OPENSPARC T1 & T2 OVERVIEW

Rick Hetherington
Chief Technology Officer
Microelectronics
Sun Microsystems

