

T2: Core Pipeline

- OpenSparc Slide Cast Ch 5: p89–94
 - each core has two (integer) execution units, 1 load/store unit, 1 FPU
 - can issue two instrns (from different thread (groups)) per cycle (≤ 2 integer/branch, ≤ 1 load/store and ≤ 1 floating point)
 - shallow instruction pipeline [p90] – simplifies core design
 - (conditional) branches are assumed not taken (no BP tables!); 5 cycle penalty if taken
 - 3 cycle penalty between load and first use
 - main pipeline 'extends' into load/store unit and FPU pipelines [p91]
 - 6 cycle latency between dependent floating point instructions [p90]
- latency (l) is high, but effect is reduced by w -way hardware threading (conversely, is amplified by w' -way superscalar)
- required 'instruction gap' between dependent instructions is $\frac{l}{w}$ (not lw' !)
 - c.f. optimized matrix multiply for $l = 3$ $w' = 4$
this complexity not needed if run fully-threaded on the T2

T2: Thread Scheduling

- hardware threading introduces a pick stage and per-thread 8-entry instrn. buffers [p91]
- the fetch stage fetches up to 4 instructions for the thread just selected in the pick stage
- threads are divided into two groups of 4; one instruction from each group may be issued [p92] – simplicity/more scalable
 - selected from available threads, using least-recently fetched policy [p93]
 - ◆ however, thread may lose priority after a load [p94]
 - ◆ e.g. Kongetira et al, Figs 4-5, IEEE Micro '05
 - thread selection is also based on resource (functional unit or register dependency) conflicts (e.g. `fmuld %f0,%f1,%f2 ; fadd %f2,%f4,%f4`)
 - thread may become unavailable due to cache / TLB miss, long latency instruction (e.g. `fdivd`), branch misprediction, trap
 - ◆ also if instrn. buffer is full!

T2: Intra-Core Memory Sub-system

- each core has a level-1 data (D\$: 8KB, 4-way set associative, 16 bytes lines, virtually indexed - physically tagged) and instrn. (I\$: 16 KB, 8-way; why larger?); shared between threads
 - D\$'s write-through policy simplifies coherency issues but increases traffic to L2\$ [p96]
 - ◆ however, it does not allocate on a store miss
 - a store buffer is used to mitigate this (can pipeline stores to the same 64-byte L2\$ line)
 - 16 byte data transfers to/from the L2\$
 - load miss queue need only support one pending load miss per thread
 - ◆ on non-CMT Sparcs (e.g. V1280 – 2003), these were 8 deep; why?
- gasket to ensure fair access to L2\$ between threads (priority to oldest)

T2: Inter-Core Memory Sub-system

- crossbar: provides high bandwidth between cores and L2\$ [p101]
 - three cycle arbitration protocol (request, arbitrate and grant); priority to oldest
 - 90 GB/s write, 189 GB/s read (8 load/store requests & 8 returns / acks / invalidations) simultaneously
- L2\$ is organized into 8 banks (address bits 8:6 used to select banks)
 - full bandwidth to L2\$ can only be achieved if each core is accessing data in a different bank on a given cycle!
 - 26 cycle latency
- L2\$ implements atomic instructions (e.g. `ldstub [%o0], %o1`)
 - semantics requires store buffers to be drained (entries for all 64 threads) 1st!
 - effectively stops most other memory operations; becomes effectively more expensive with increasing number of cores (20-30 cycle latency)
- L2\$ manages coherency via separate directories to all 16 I\$ and D\$ [p102]
 - broadcasts an invalidate signal whenever a value is changed
- memory is organized into 4 banks (selected by address bits 8:7) [p86,102], enabling 42 GB/s read, 21 GB/s write; s/w can prefetch data into the L2\$

UltraSPARC T2: Speculative Execution

the ultimate instruction level parallelism technique in high power/chip area costs and low on average (single threaded) performance gains, so ...

UltraSPARC T2: Power Management

CoolThreads: low cooling requirements

p99 CPU: worst case power usage 84W (1.4 GHz); T5120 system: 330W

- can turn cores, DRAM banks on/off via software
- H/W uses clock gating to turn off functional units (e.g. FPUs) not in use
- power throttling: can inject stall cycles to cool chip, or conserve power in low workload periods
- improved reliability with lower and more uniform junction temperatures, on-chip temperature control

OpenSPARC T2: A Open-Source Chip Design

- see www.opensparc.net/opensparc-t2
- includes micro-architecture specification, RTL and verification suite
- supports projects in
 - create new (variant) core designs
 - explore CMT design points for new applications
 - port OS of your choice, explore program/compiler optimizations
 - using software (simulator) and/or single core single thread FPGA implementation of OpenSPARC T1

UltraSPARC T2: Performance

- world record SPECint_Rate2006, SPECfp_Rate2006 (single chip) [p85]
 - not optimized for FP; and a low clock speed?
 - for dual-socket systems (Kumar et al, 2008):

	clock	cores	f.p/cycle	R_peak	SPECfp2006
T5240	1.4 GHz	16	1	22.4 GF	119
IBM p570	4.7 GHz	4	4	75.2 GF	116

c.f. 3.2GHz Cell has R_peak of $3.2 \times 8 \times 4 = 102$ GF

- reason: FLOPs aren't everything; memory performance is important!
- world records (single or dual chip) on SPEC web, Java VM, Java App, OLTP etc
 - for SPECweb2005, T5220 rating: 41847 (@426W)
 - nearest: 2×4 -core Opteron Sun Fire X4240: 32288 (@411W)
- niche is cryptographic applications (SPUs act as on-board accelerators)
 - e.g. out-performs 2.7GHZ quad-core Clovertown by factors of ≈ 10

UltraSPARC T2: Design for Performance Choices

- aggressively multi-core and threaded design
 - avoid expensive single-threaded motivated features:
 - ◆ branch prediction, (wide) superscalar, O-O-E, deep pipeline, speculative ex. in favor of a simpler, highly threaded approach
 - ◆ can afford higher I & lower GHz; reduced s/w complexity (once threaded); no need for deep prefetch buffers; can better tolerate cache misses **but:**
 - ◆ pick stage, per-thread instrn. & store buffers, thread scheduling (reduced by 2 sub-groups), many sets of register files, larger and more associative caches, coherency and atomic instruction implementation
 - equal effort put into high bandwidth L2\$ and main memory access
 - ◆ banking introduces complexities (gasket & crossbar) in h/w (& optimizing s/w)
- could be (easily) improved for floating point (dual FP issue or fused multiply-add), but intended for other markets (memory & I/O intensive; encryption)
- supporting full SPARC ISA somewhat of a burden
 - c.f. Terascale chip (80 cores @ 4.3GHz; 32-bit multiply-adder; minimal ISA, on-chip memory & I/O; 97W)
- power/performance also important (low peak power & power saving features)