A 32-way Multithreaded SPARC® Processor

Poonacha Kongetira

Authors

• Kathirgamar Aingaran
• Les Kohn
• Poonacha Kongetira
• James Laudon
• Kunle Olukotun
• Sanjay Patel
• Ramaswamy Sivaramakrishnan
• Sunil Vemula
Agenda

• Niagara Threading Model

  – Workload Characteristics
  – Niagara Approach
  – Threading Implementation
  – Providing Memory Bandwidth for 32 threads

Commercial Server Workload Characteristics

• High degree of thread level parallelism (TLP)
• Large working sets with poor locality of reference leading to high cache miss rates
• Significant data sharing among threads resulting in coherence misses
• Low Instruction Level Parallelism (ILP) due to high cache miss rates, load-load dependencies, difficult to predict branches
• Performance is bottlenecked by stalls on memory access

Difficult to achieve higher performance in these applications efficiently with superscalar and superpipelined methods!
ILP processors mostly reduce CPU time, but memory stall time dominates overall performance!

Larger number of Memory References outstanding from overlapping h/w threads leads to higher throughput
Niagara Approach

• CPU with 32 threads exploits TLP
  – Memory and Pipeline stall time hidden by multiple threads
  – Shared L2 cache allows efficient data sharing among threads
  – Shared pipelines allow reuse of resources by overlapping threads, thereby saving area

• Memory system designed for higher throughput
  – High bandwidth interface to L2 cache for L1 misses
  – Banked and highly associative L2 cache
  – High bandwidth interface to DRAM

• Performance/Watt as a design metric

32 Threaded Niagara CPU

• An implementation of SPARC V9 architecture.
• Eight 64-bit 4 way multithreaded pipelines
• 4 way banked 3MB secondary cache
• High bandwidth crossbar interconnect for on chip communication.
• High bandwidth DRAM interface
SPARC Pipe – Salient Features

• Single issue pipeline
• 4 threads supported per pipe
• Each thread has unique register state, instruction buffers and store buffers
• Caches, TLBs, execution units, and pipeline resources are shared

SPARC Pipe – Basic Dataflow

Fetch → Thrd Sel → Decode → Execute → Memory → WB

ICache Inst buf x 4

Thread selects

Decide

Regfile x4

Instruction type
misses
traps & interrupts
resource conflicts

Thread select logic

Alu Mul Shft Div

DCache Dtlb Stbuf x 4

Crossbar Interface

Thrd Sel Max

PC logic x 4
Thread Selection Policy

- Switch between available threads every cycle giving priority to least recently executed thread
- Threads become unavailable due to:
  - Long latency ops: loads, branch, mul, div.
  - Pipeline stalls such as cache misses, traps, and resource conflicts
- Loads are speculated as cache hits, and the thread is switched in with lower priority
Thread Selection – Two Threads Ready

Cycles

Instructions

- \( S_{t0-ld} \)
- \( D_{t0-ld} \)
- \( E_{t0-ld} \)
- \( M_{t0-ld} \)
- \( W_{t0-ld} \)
- \( F_{t0-add} \)
- \( S_{t1-sub} \)
- \( D_{t1-sub} \)
- \( E_{t1-sub} \)
- \( M_{t1-sub} \)
- \( W_{t1-sub} \)

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

L1 Caches

- Level 1 caches are shared among 4 threads
- I Cache: 16kB, 4 – way set associative, 32B line size
  - Fetches 2 successive instructions each cycle
  - I cache access for instruction fetch is not required every cycle. Open slots are used to do fills without affecting fetch bandwidth
- D Cache: 8kB, 4 – way set associative, 16B line size.
  - Writethrough cache, allocating loads, non allocating stores
- Store buffers: 8 entry buffer per thread
  - Threads only lookup their own store buffer
  - RAW bypass supported within a thread
Register File

• 3R/2W ports, supports 8 register windows and 4 threads.
• In the SPARC register window implementation, only 3 register windows (working set) can be seen at a given time.
  – The architectural set of registers is implemented using 6T srams. A transfer port links the arch set to the working set of registers.
  – The working set of registers (32) is implemented using fast register file cells
  – Window changing operations switch out a thread allowing data to be transferred between working and arch registers.
  – Only one thread can read the RF at a time, therefore cells for the 4 threads are merged into a compact structure with shared read bitlines.

Crossbar

• High b/w interconnect between SPARC pipes, cache banks and other shared resources
• Non blocking design – allows multiple transactions to be queued up from a source
• Age order dispatch to destinations ensures fair scheduling and no starvation
• Provides ordering point for memory transactions
L2 Cache

• Designed for high bandwidth and low power
• 4 way banked 3MB L2 cache
• 12 way set associative to handle 32 threads
• Data is 64B interleaved across banks
• 64B block size
• Multiple outstanding miss handling
• Writeback protocol

Cache Coherence

• L2 cache is shared by 32 threads
• L1 cache is per core and shared among 4 threads per core
• L1 lines have 2 state: valid or invalid
• L1 is writethrough – no modified state
• L1 caches kept coherent by tracking L1 lines in directories kept in L2
• When a memory location is modified, invalidates are issued by the L2 based on the
L2 Cache – Directory

- Reverse mapped directory implemented in L2
  - Tracking done by L1 line instead of L2 line by shadowing the L1 tags
  - Ratio of no of lines of L1 to lines of L2 make this more efficient, even though it is a cam structure
  - Indices are arranged so that stores cam against 1/16th of the directory making this very power efficient

DRAM Access

- Multiple DDRII DRAM channels
- Supports memory size of upto 128GB
- Excess of 20GB/s peak bandwidth
Conclusions

- Commercial Server Workloads exhibit high degrees of Thread Level Parallelism, but poor locality of reference
- 32 threads are implemented on a single CPU to hide memory and pipeline stalls and maximize parallel memory accesses
- A high bandwidth memory subsystem with shared cache, services memory references
- Sharing of resources at all levels leads to a very area and power efficient design