POWER4 System Architecture in conjunction with POWER4 Circuit and Physical Design of the POWER4 Microprocessor

Natalie Enright And Dana Vantrease

Thursday, September 25, 2003

1 Quickfacts

1.1 Processor

- 2 cores/processor
- Speculative SuperScalar Out-of-Order
- On-chip: L1 Icache, L1 Dcache, L2 cache, and L3 cache controller
- Off-chip: L3 cache
- Seperate Issue Queues
- 8 Execution Units
- 1.1-1.3 GHz
- 174 million transistors

1.2 System

- 2 processors/chip x 4 chips/module x 4 modules/SMP = 32-way
- "distributed switch" interconnection network

	Associativity	Line Size	Capacity/chip	Inclusion?	Coherency States
L1 Inst	Direct Map	128 bytes	128 KB	Yes, w/ L2	Invalid, Valid
L1 Data	2-way	128 bytes	64 KB	Yes, w/ L2	Invalid, Valid
L2	8-way	128 bytes	$\sim$ 1.5 MB		Invalid, Shared local, Shared, Modified, Exclusive,Unsolicited, Modified, Tagged
L3	8-way	512 bytes	32 MB	No	Invalid, Shared, Tagged, Remote Tagged, Prefetch

2 Interesting Design Decisions

2.1 ERAT (E.ffective to R.eal A.ddress T.ranslation)

The ERAT table, which comes in two flavors – instruction (IERAT) and data (DERAT) — stores the TLB translation of an effective address to a real address. It is 128 entry, 2-way associative, compared to the TLB, which is 1024 entry and 4-way associative, and thus provides quicker translation. Introducing an ERAT to each processor gives lookup a hierarchical structure, similar to that of caches.

2.2 Grouping Instructions

Instructions are tracked through the system in groups of up to 5. When an instruction group is formed and ready, it is dispatched. Upon completion of each instruction in the group, the group is retired and the state is updated on the group boundary. While groups are formed in order, instructions within groups are allowed to execute out of order. Groups may also complete out of order but must retire in order. A group must wait in the Group Completion Table (GCT) until all members of its group and of the preceding groups have completed. Worth noting is the point that there are plenty of rules placed on the dispatching of groups. To name a few:

- Slot 4 of the group is reserved only for branches
- Only one group may be dispatched per cycle

- Only one group may be completed per cycle
- Instructions reading more than 2 registers and/or writing more than one must be either cracked (1→ 2 ins) or millicoded (1 → 3+ ins) Cracking allows the compiler/programmer to specify CISC-like instruction while maintaining a RISC-like pipeline. Cracking also simplifies scheduling and reduces register port requirements.
- If an interrupt occurs within a group, the group must be flushed and redispatched in single instruction groups to isolate the interrupting instruction. The POWER4 has a significant and complicated rollback and replay mechanism, which is completely left out of the paper. This elaborate replay mechanism allows the POWER4 to do a lot of speculation.
- To be dispatched, all of the following must hold or else the group is stalled:
  - A Group Completion Table entry must be available (it tracks the group)
  - Respective Issue Queue slots must be available for each instruction
  - Adequate Rename Registers must be available and assigned
  - If loads exist, L.oad R.eorder Q.ueue entries must be available
  - If stores exist, S.tore R.eorder Q.ueue entries must be available

Looking over the restrictions placed on groups, it becomes obvious that a single bad instruction can really spoil the bunch. The complex branch prediction scheme, with its local, global, and selector prediction tables, tries to curb this problem by avoiding unnecessary bubbles in the pipeline.

Separate Load and Store queues allow for faster searches for forwarding values or detecting conflicts.

2.3 Interconnection Network

Each chip has a bus spanning the module that serves to communicate intra-modularly. Also, each processing node has a ring bus going through it that connects it to its respective processing node in every other module to communicate inter-modularly.

When a chip writes, it writes to its own bus. When the communication reaches the ring, it travels on one of four rings, giving the impression that the network looks like a switch from the module's perspective. However, any

transaction cannot go on any ring. The transaction must travel on the ring associated with its chip.

Meanwhile, the bus network looks like a bus from the chip's perspective because though the request comes in one chip, it is distributed throughout the module. Finally, the ring structure also helps maintain memory consistency because it can infer some ordering information about the passing transactions since transactions are not allowed to pass each other as they go around the ring.

3 Discussion

3.1 How much does grouping help/hurt?

Hurt:

- Risks choking dispatch bandwidth if GCT prematurely fills because groups are mostly empty (Internal Fragmentation).
- Less granularity due to group boundary-kept state of instruction-kept state. Also, interrupting instructions and their groups must be dispatched twice to adjust granularity to single out interrupting instruction.
- Less flexibility due to loss of control over dispatch and instruction issuing

Help:

- Scalability. Less logic is required to support the pipeline because there are less groups (up to a factor of 5) than instructions to track. Also, the smaller Group Completion Table is faster to access.
- Exploit ILP with cracking/millicoding
- Possibility for increased help with compiler support:
- Use the freedom to rearrange independent instructions towards forming groups on 5 instruction boundaries around branches.
- Schedule instructions such that they balance the separate issue queues that feed duplicated execution units. This is important since the queue that instructions issue to is solely dependent on their location in their group.

3.2 Why are cracked instructions allowed to be in the same group and millicoded instructions must start a new group?

To maintain atomicity on group boundary state updates, cracked ($1 \rightarrow 2$ ins) instructions can fit adequately in one group and thereby maintain precise exceptions and atomicity. An example of such an instruction would be a load and increment register instruction. However, millicoded instructions ($1 \rightarrow 3+$ ins) are larger and may span group boundaries, affecting the atomicity illusion. For example a load multiple word instruction can be a millicoded instruction but it is not atomic. Fortunately, it is also very infrequent.

3.3 Why is the L2 implemented as three slices?

We speculate there was only room on the chip for three slices. Also, someone in class contributed that they believed not all of the slices were entirely 8-way, and that because of size limitation, some sets had less than 8 ways.

3.4 L1, L2, L3 coherency protocol

The coherency protocol at the L2 level is a modified MESI protocol with 7 states. The inclusion bit is unique to this architecture which can be helpful for Multiprocessor (MP) systems. However, the inclusion bit is not precise. It gets set in the L2 cache when the L1 cache requests a line however it does not get reset when the L1 cache casts the line out. This bit is useful because it potentially reduces the number of invalidations that need to be sent to the L1 cache in a MP system. If the line is prefetched into the L2 then the inclusion bit will not be set and a subsequent invalidate will terminate at the L2 cache, reducing bandwidth demands between the L2 and L1.

3.5 There was a lot of discussion about the specifics functioning of the L3 cache. Does the L3 cache act as a memory side-cache? Can the L3 cache only addresses contained within the memory it is attached to?

The conclusion reached in class was that the L3 cache could cache any data and was not restricted to the data contained in its memory. The Trem (Tagged Remote) state in the L3 coherence protocol supports this conclusion

since it indicates that data was sourced from the memory attached to a different chip.

3.6 What is memory scrubbing?

This has to do with ECC. Don't let latent single bit errors just sit there; periodically read out and write back memory to correct the error.

3.7 Looking Forward: What are the pros/cons of selling a CMP system with spare processors with the intention that they could be hot-swapped out if one of the originals failed? Alternatively, how about using all processors in the system and simply turning off a bad processor?

A mainframe IBM did this with having a spare set of processors that used the L2 state of the dead processor in recovery

Pros:

- When one processing core is bad, the whole system is not
- Easy Recovery

Cons:

- Coherence protocol must accommodate for change/loss of processors
- Complex Hot-swapping logic
- Must protect against users using spare processors in addition to their original allotted number of processors.

3.8 Why is the L3-directory on-chip and the L3-cache off-chip?

The check for hit/miss in the L3 can be done on-chip versus off-chip, resulting in lower latency for cache-to-cache transfers. Fast cache-to-cache transfers are important for obtaining high throughput in commercial workloads, as listed in IBM's "Guiding Principles". Finally, putting the logic on chip means fewer L3 queries going off-chip and taking up valuable bandwidth. Since the L2 and L3 do not maintain inclusion, the L2 controller on chip will need to be checked as well as the L3 directory so having them both on chip reduces off chip queries.

3.9 Why make the L1 maintain inclusion with the L2?

Note: Piranhas size(L1) =size(L2) and does not have inclusion

- L2 acts as a backup copy of data in L1. This can be incredibly useful in restoring state when processing nodes are hot-swapped.
- Allows the L1 to go faster and require less ports
- Cache-to-cache transfers are faster because the requests can stop at the L2.
- Passes coherency control to the L2

4 Notes from circuit paper

The POWER4 has a very hierarchical design strategy organized into macros, units, core and chip. The design was achieved with primarily static circuits. This was done to cut down on power dissipation and to ensure a simple and robust design at the transistor level. Design for testability was also a major focus in this paper. They used their level sensitive scan design (LSSD) to ensure testability of the chip. This can test timing as well as many other faults.

Included in the design of the caches is word and bit line redundancy. Word and bit line redundancy consumed a small amount of extra area but improves yield tremendously since it guards against an open or a short faults, which are common in manufacturing. IBM is much more quality oriented than Intel.