Comparative Architectures Supervision Answer (Set 4)

Wei Song 17/04/2016

This answer provides a summary of key points in each question. It is the understanding of the supervisor, which does not necessarily always correct and does not represent the view of lecturers. It is expected to expand the key points into detailed description when similar questions are asked in exams.

Lecture 12 & 13. Chip multiprocessor

Q1. Why is a shared second-level (L2) cache typically divided into multiple banks (banked) in a chip multiprocessor?

Banked L2 allows multiple simultaneous cache requests accessing different banks.

Q2. In what situation might a shared second-level cache offer a performance advantage over a memory hierarchy for a chip multiprocessor with private L2 caches?

- A shared L2 cache reduces the duplicated caches lines needed when using private L2 caches.
- A shared L2 cache allows a thread to occupy more than its share of L2 (but may cause cache thrashing)
- However a shared L2 normally prolongs L2 access latency.

Q3. A cache controller in a chip multiprocessor snoops the bus and observes a transaction that refers to a block that its cache contains. The block is held in State M (Modified). The bus transaction has been generated by a processor wishing to read the block. Assuming a MSI (write-back invalidate) cache coherence protocol, what actions will be taken by the cache controller?

The cache controller writes this block to memory if it is dirty and provides the data to the processor reading this block. The status of this block changes from M to S.

Q4. How does adopting an inclusion policy simplify the implementation of a cache coherence mechanism in a chip multiprocessor with private L1 and L2 caches?

So a snoop on a shared bus needs to check only the tags of L2.

Q5. How might multiple buses be exploited to enable a greater number of processors to be supported by a snoopy cache coherence protocol?

Similar to banked L2, multiple buses allow parallel cache transactions using separate buses to increase throughput and reduce latency.

Q6. Why might we use a directory-based cache coherency protocol in preference to a snoopy protocol?

When the cost of snoop is high, such as the cost of putting all cores on a single bus, the cost of broadcasting and the cost of requiring a cache controller to constantly monitor all bus transactions, a directory-based cache coherency protocol may provide a scalable solution.

Q7. What optimisation does the addition of the E state to the MSI protocol provide?

Reduce the request to change state from S to M when a write is needed.

Lecture14. On-chip interconnect

Q8. For what reasons might virtual-channels be added to an on-chip network?

- Increase network throughput.
- Support QoS.
- Avoid deadlocks.

Q9. Why might an adaptive routing algorithm offer better performance than a deterministic one?

Some adaptive routing algorithms may guide traffic to reduce network congestions; therefore, increase the overall network throughput. However, adaptive algorithms do not always outperform deterministic algorithm. This benefit is network and traffic dependent.

Some adaptive algorithms are tolerant to certain amount of faults which improves network reliability.

Q10. How might the choice of cache coherence protocol influence the design of the on-chip network?

- Topology: bus, tree or mesh, depending on constant or variable access latency.
- Number of virtual channels or network slicing (parallel networks): avoid deadlock and support multiple simultaneous transactions.
- Network throughput: to support the average or worst case requirement for memory bandwidth.
- Routing algorithm: point to point or multicast, deterministic or adaptive.

Lecture15. Special-purpose architectures

Q11. How can performance be improved while reducing power consumption?

The answer for this question is really open. There are lots of possibilities.

- Special coprocessor or hardware allows fast and low-power operation of special tasks.
- Simple in-order or short pipelined cores is better than deep out-of-order cores for simple, low-performance or single-thread tasks.