**Worksheet 10**

**Q1. MOESI Coherence**

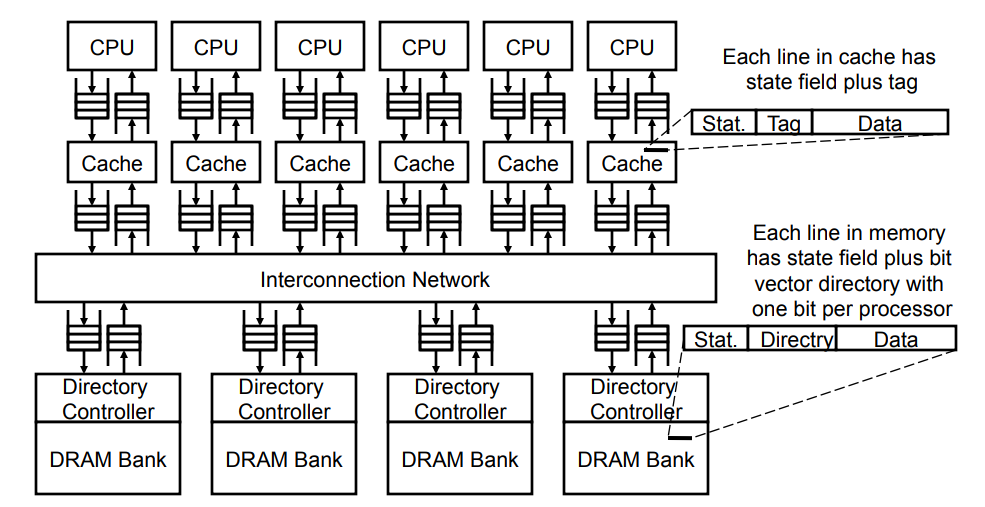
MOESI is an improvement on the MESI cache coherence protocol that adds a fifth “Owned” state which allows caches to hold dirty data without invalidating sharers. Only one cache can be in the “Owned” state while the other caches are in “Shared”. The owner is responsible for sending data to other caches requesting the line and must write back the data when it downgrades.

For each of the following new state transitions, indicate whether it is a valid transition. If it is a valid transition, explain what triggers it, what conditions must be true (i.e. do other sharers exist?), and what actions must be taken during the transition.

|  |  |
| --- | --- |
| I → O |  |
| O → I |  |
| S → O |  |
| O → S |  |
| E → O |  |
| O → E |  |
| M → O |  |
| O → M |  |

**Q2. Directory Cache Coherence**

In lecture, the professor mentioned that directory cache coherence protocols can scale up more easily than snoopy cache coherence protocols. Let’s examine how such a system scales as we drastically increase the number of CPUs in the system.



In the simplest design, each directory entry contains the state of the cache line and a bit vector indicating all of the sharers. Assume the directory lines have four states.

1. How many bits does the directory need to store per cache line if there are 128 CPUs, each with its own private cache?
2. How many bits does the directory need to store per cache line if there are 1024 CPUs?
3. It should be obvious from the previous calculation that storing one bit for each sharer per line will not be practical for massively multicore systems. For the 1024 core system, if cache lines are 64 bytes in size, you will end up using twice as much memory for the sharer bits alone than you do for actual data storage.

We therefore decide to collect processors into groups and save each group’s state in a single bit in a group vectors. Invalidations must now be sent to all of the CPUs in a group if that group’s bit is set.

For a 1024 core system with 64-byte cache lines, how many cores must be in each group to reduce the amount of directory state to 10% the amount of memory?

1. One inefficiency of this system is that you must store directory bits for every line in memory, no matter if it is cached or not. How could you reduce this inefficiency?