Clarifications for COL216 Assignment 3

Clarifications

Query 1. Write Conflict Timing and Tie-breaking

Question: If two cores issue writes to the same address at the same time, how is contention resolved? Does the losing core retry or get queued?

Clarification: Only one bus transaction per address can proceed at a time, the others must wait. For example, if Core 1 writes to address 0x0 in cycle 0, Core 2's write request generated simultaneously loses the bus arbitration. Core 2's request is not queued or remembered, unless you explicitly implement such a queue. Instead, it reissues the write request in the next cycle, that is, it retries until it gains access. Core 1 will continue to issue its next instruction only after the current one completes, as caches are blocking. Tie-breaking is arbitrary, you may use core index as the default order. So yes, Core 2 will retry the request in subsequent cycles until granted.

Query 2. MESI State Transition Timing from Modified to Shared

Question: When does the sender's MESI state change, at the start or end of the transaction?

Clarification: Suppose Core 0 has a block in the Modified state and Core 1 issues a Read. The bus transaction begins in cycle 0. Core 0 responds by sending the block, which takes 2N cycles. During this time, Core 0 may update its state from Modified to Shared at the start of the transaction, that is, in cycle 0. Core 1 will update its state to Shared only at the end of the transaction, after the 2N cycles have passed. This follows the MESI rule that both caches hold a clean, shared version of the data only after the transfer is complete.

Query 3. Bus Usage During Block Transfers

Question: How long is the bus busy during cache to cache or memory transfers?

Clarification: For cache to cache transfers, the bus is busy for the entire 2N cycles required for the transfer. During this period, other cores cannot initiate any bus transactions. For memory fetches, the bus is considered busy for the full 100 cycles it takes to access memory. You may assume that other cores with bus requests will have to wait for the bus to become free. However, cores performing cache hits or non-conflicting operations do not need to stall.

Query 4. Cycle-by-Cycle Behavior of a Read Miss

Question: What events occur in each cycle when a read miss happens?

Clarification: Consider the following cycle breakdown when a read miss occurs at cycle T by Core 0:

- Cycle T: Core 0 issues a BusRd. Other caches snoop to check for the block.
- If the block is found in another cache, a cache to cache transfer of 2N cycles begins.
- If not found, a memory fetch of 100 cycles begins.
- Cycle $T + 1$ onward: Data is in transit, Core 0 remains stalled.
- Cycle $T + 2N$ or $T + 100$: Block arrives, Core 0 installs the block, updates MESI state, and resumes execution.

Snooping and initiating the bus transaction happen in the same cycle as the miss is detected. The requesting core remains halted until the data is received and installed.