ECE 506 PROJECT 2 REPORT

Pbnagar - 200605953

1. Optimization that MESI Does Reduce Memory Transactions

The MESI (Modified, Exclusive, Shared, Invalid) protocol reduces memory transactions through its state management, particularly the Exclusive and Modified states:

- **Exclusive State**: Exclusive state will be when the cache has the most recent copy of the data where no other caches contain the data. As it will be seen, this rules out a memory writeback when the line is modified. Additional writes can be made without involving the bus, to the device.
- **Modified State**: Modified state in cache line means that it has been written but not in the main memory but only to the cache. It put off writebacks to evict the cache line, or invalidate it, this has the effect of reducing memory transactions.
- **Cache-to-Cache Transfers**: If a processor reads data changed by another processor, the protocol enables the modified cache to hand over the data without going to main memory.

By the introduction of these states, MESI reduces unnecessary writers to memory and guarantees that data coherency is obtained without constantly referring to main memory.

2. Optimization that MOESI Does Reduce Memory and Bus Transactions

The MOESI (Modified, Owner, Exclusive, Shared, Invalid) protocol builds upon MESI by adding the Owner state, which optimizes both memory and bus transactions:

- **Owner State**: This state enables a cache to forward data to some other cache (cache-to-cache transfer) and it remains solely responsible for writing back to memory whenever the line is replaced. This reduces:
- **Bus Transactions**: Cache-to-cache transfers relieve the memory board from a read request from another processor requesting data when the Owner cache have it.
- **Memory Transactions**: The writeback of new data to memory in Owner state does not occur concurrently as a new block is written directly into memory and everything is written back only when eviction occurs.

DT2 data is made available to others by a processor that favors the retain write data and supplies it changed to others rather than going for the bus this makes MOESI more efficient with fewer busloads as compared to MESI.

3. High-Level Details of Implementations

MESI Implementation

- **State Transitions:**

  A BusRd (read request) transitions the line from Modified or Exclusive to Shared state, and its purpose is to make certain that the value that was changed is up-to-date and identical to the one written.

  A BusRdX or BusUpgr request itself, in a shared access, declares other copies shared, to be no longer valid.

- **Processor Access:**

  Read hits <access greater than 0> provide access to hits Exclusive, Shared, or Modified state without having to perform memory transactions.

  Writing hits results either in Modified or Exclusive state to avoid bus transactions thus enabling efficient local updates.

  On misses, data is gotten from memory, or another cache and it is in Modified state.

- **Bus Snoop Logic:**

  Observes bus transaction to manage transition of state and coherence.

MOESI Implementation

Enhancements Over MESI:

Introduces the Owner state that enables a cache to deliver data to other caches without appropriate memory.

It is still possible to share Daten in the Owner State while at the same time letting one cache take care of memory writebacks.

Processor Access:

Read hits in the Owner state give information locally without even referencing memory.

Avoid memory or bus transactions on hits to Modified or Owner states write.

Cache-to-cache transfers even further minimize cross traffic on the bus and memory.

Bus Snoop Logic:

Extra logic to the Owner state transitions to facilitate data transfer and memory synchronization also forms part of the architecture.

Performance Considerations

Latency: The implementation measures the clock cycles of hit and miss latencies where necessary to emulate memory and bus cycle delays.

Counters: For the assessment of the protocol performance, counters such as read hits, write hits, flushes, invalidations are used.

Simulation: The implementation imitates cache behavior for several cores to achieve interconnect MESI and MOESI protocols.

These implementations effectively, accurately and optimally mimic actual cache coherence protocols by reducing memory and bus traffic.