Lecture 30 (Cache Coherence)

1 Write Update Protocol

1.1 States

1. M - modified
2. S - shared
3. I - invalid

1.2 Event/Message Notification

1. Rd - Read request
2. Wr - Write request
3. Evict - Evict the block
4. Wb - Write back data to the lower level
5. Update - Update the copy of the block

1.3 Message Types

1. RdX - Generate a read miss message. Send it on the bus/NoC if required
2. WrX -Generate a write miss message. Send it on the bus/NoC if required
3. WrX.u - Get permission to write to a block that is already present in the cache
4. Broadcast - Broadcast a write on the bus
5. Send - Send a copy of the block to the requesting cache

1.4 Snoopy Protocol

1. All messages are broadcast messages
2. Since they are sent on bus, all caches can read
3. Easy to design but not scalable

1.5 State Machine

1.6 Events Received from the Bus

1. Only one sister cache can use the bus at any time, global order of writes is preserved

2. If bus master disallows starvation, then all writes will complete

2 Write Invalidate Protocol (MSI Protocol)

1. In M state, only one cache can contain a copy of the block
2. Multiple cache can have the block in S state as read-only

2.1 State Machine

Such state machines are called "Distributed State Machines" or "Actor Models"

3 MESI Protocol

1. Introduce a new state E - exclusive
2. Avoid modify request on bus when exclusive owner

4 Important Questions

1. Who supplies data if sister sends read miss or write miss?
   - Caches who have a copy of cache, arbitrate for the bus. Whoever gets the access first, sends the data. Others snoop this and cacnel their request.
   - This leads to overhead
2. Do we need to write-back on $M \rightarrow S$ transition?
   - If we can avoid it, then we can optimize performance and power

5 MOESI Protocol

1. Introcude O - owner state
2. This state will send data for any read miss requests
3. It write-backs on eviction
4. We also introduce some temporary states in case there is no owner