**ECE382N.10 Parallel Computer Architecture Lab 2**

**Work division**

**Qiyang Ding: Protocol design and test design/implementation**

**Mingyu Lei: Protocol design and implementation**

1. **Part 1: Warm Up**

Since the random here is uniform distribution and cache size is half of the memory size, the hit rate should be 0.5. However, the stats consider full hit and partial hit based on load and store operation and hit rate is calculated only by full hit. Therefore, the hit rate is less than 0.5. To fix this problem, there are two ways to solve this. One is to directly store the data into the cache line once it hits because cache coherence is not considered in this case. The other is to increase the cache size to give more hits.

1. **Part 2: RWITM**

The difference between storing into an invalid cache line and a shared cache line is the data requirement. Therefore, if they are treated the same, there are less logic to determine whether the cache line is shared or valid. The normal way is to separate them into two operations because shared cache line can save one data request from the bus.

1. **Protocol Design and implementation**

There are four states in the directory entry, which are Invalid, Shared, Owned, and Shared-no-data. Invalid means that no one shares or owns this cache line; Shared means someone has the copies of this cache line; Owned means one node owns this cache line; Shared-no-data means this cache line will be written back from the owner, but it is not ready.

There are two new network queues (priority) called Writeback and Forward. Writeback queue is used for optimizing the transaction speed of writeback request while forward queue is used to separate normal requests with forward requests to avoid deadlock.

There are two internal buffers for invalidation requests and other requests respectively. The size of invalidation buffer is 32 (requests) and send-request buffer is 2 (requests). If there are any pending request in the send-request buffer, all the directory operation will be blocked.

Processor command from cache to IU will be only one per cycle. If there are replacement or snoop operation during current cycle, the one from cache load/store will not be executed any more.

* 1. **Directory and Requests**

|  |  |  |  |
| --- | --- | --- | --- |
| Req Type Dir State | Read | RWITM | Write-back |
| Invalid | If there are no sharers,  Dir -> Owned   1. Update sharer list and owner 2. Send reply to the requester with data   Else,  Dir: Invalid  Reply to the requester with non-acknowledge | If there are no sharers,  Dir -> Owned   1. Update sharer list and owner 2. Send reply to the requester with data   Else,  Dir: Invalid   1. Reply to the requester with non-acknowledge | X |
| Shared | Dir: Shared   1. Update sharer list 2. Send reply to the requester with data | Dir -> Owned   1. Send invalidation requests to all the sharers 2. Update sharer list and owner 3. Send reply to the requester with data | Dir: Shared   1. Update sharer list |
| Owned | If sharer list shows that other nodes except for the owner have the copy,  Dir: Owned   1. Reply to the requester with non-acknowledge   Else,  Dir -> Shared-no-data   1. Forward the request to the owner | Dir: Owned   1. If the requester is one of the sharers, reject the request to the requester 2. If the requester is not one of the sharers, forward the request to the owner | Dir -> Invalid   1. Write data into related memory 2. Update sharer list and owner |
| Shared-no-data | Dir: Shared-no-data   1. Forward the request to the owner | Dir: Shared-no-data   1. Reject the request to the requester | Dir -> Invalid   1. Write data into related memory 2. Update sharer list and owner |

* 1. **Directory and Replies**

|  |  |  |  |
| --- | --- | --- | --- |
| Req Type Dir State | Forward Read Reply | Forward RWITM Reply | Invalidation Reply |
| Invalid | X | X | Dir: Invalid   1. Update sharer list (Change 1 to 0) |
| Shared | X | X | Dir: Invalid  Update sharer list (Change 1 to 0) |
| Owned | Dir -> Shared   1. Copy the data into local memory 2. Update sharer list | Dir: Owned   1. Update the owner id in the directory entry | Dir: Invalid  Update sharer list (Change 1 to 0) |
| Shared-no-data | X | X | Dir: Invalid  Update sharer list (Change 1 to 0) |

* 1. **Cache and Requests**

|  |  |  |  |
| --- | --- | --- | --- |
| Req Type Dir State | Forward Read | Forward RWITM | Invalidation |
| Invalid | Cache: Invalid   1. Reply to the requester with non-acknowledgement | Cache: Invalid   1. Reply to the requester with non-acknowledgement | X |
| Shared | Cache: Shared   1. Reply to the requester with data | X | Cache -> Invalid   1. Invalidate related cache line 2. Update replacement status of the set |
| Exclusive | Cache -> Shared   1. Reply to the requester with data 2. Reply to related directory with data | Cache -> Invalid   1. Reply to the requester with data 2. Reply to related directory with data | X |
| Modified | Cache -> Shared   1. Reply to the requester with data 2. Reply to related directory with data | Cache -> Invalid   1. Reply to the requester with data 2. Reply to related directory with data | X |

* 1. **Cache and Replies**

|  |  |  |
| --- | --- | --- |
| Reply Type Dir State | Read Reply | RWITM Reply |
| Invalid | Cache: Based on permit tag (Shared, Modified, Exclusive)   1. Find related set in the cache 2. Fill the cache (May cause replacement) 3. If there is replacement occurred, send the request to the IU for write-back | Cache -> Modified   1. Find related set in the cache 2. Fill the cache (May cause replacement) 3. If there is replacement occurred, send the request to the IU for write-back |
| Shared | X | X |
| Exclusive | X | X |
| Modified | X | X |

1. **Test infrastructure**

Test infrastructure is built based on four vectors. They are test case vector, test record vector, test result vector, and test golden vector. The first one is to control the processor behavior (load/store) based on specific cycles. The second one is to control the processor to record the internal state in directory and cache. This vector will be only used in microarchitecture tests since it needs specific interface of cache and directory. The third one and fourth one are the internal state/data and expected state/data. It will automatically compare the results with golden and show errors depending on different values.

1. **Test cases**

There are many microarchitecture cases that compare the internal state and its strict cycles. These tests include four tests for invalid directory state (local and global), seven tests for owned directory state (local and global), twelve tests for shared directory state (local and global), five tests for shared-no-data directory state (local and global), three tests for checking invalidation queue, and one random test.

There are also many assertions in the codes to keep internal state as expected and they are also used in random test to test the stability of the whole system.