Project Report ECE- 585

Topic: Simulation of Last Level Cache (L-2) using C programming Language

Under Guidance of **Prof. Mark Faust**

Team: 5

Abhiraj Abhaykumar Eksambekar	**abhiraj@pdx.edu**
Aniruddha Shahapurkar	**anirud@pdx.edu**
Deepen Parmar	**parmar@pdx.edu**
Disha Shetty	**dshetty@pdx.edu**

- Specification of Cache

1) Total Capacity of the Cache = 16Mb
2) Capacity of each line = 64 bytes
3) Way = 8
4) Policy = write Allocate
5) Protocol used for cache coherence = MESI
6) Replacement Policy = PLRU

- Calculations

7) Address bits = 32
8) Byte Select bit = 6 bits
9) As it uses 64 bytes/line, so it needs 6 bits for byte select that is 6 lower bits of the address.
10) Number of cache lines = Total Capacity of Cache / Capacity of each line = 16mb/64 = 256K
11) Number of Sets = 256K/8 = 32K
12) Index = 15 bits
13) Tag bits = 32 -15-6 = 11 bits

- Assumptions:

  In L1 we assume that, whenever L1 sends write request, it sends data with it and therefore we modify our line with the given data. But L1 might be doing Flush operation due to eviction or First write through operation or Write miss and we can't predict the reason for write. So, we always send requested line to L1 assuming that if L1 has write miss, it will take the line; if it is first right through or flush it will ignore the line.

- Operations:

  1) read_request_from_L1_data_cache:

     When there is read request from L1 cache

     Case A: When address is present in L2 cache:

     (i) Check if that address is present in L2 cache using search_cache() function. When it is present (it will be a hit), send the message to the cache, that line is present and you can read it.
     (ii) Update the PLRU bits.

     Case B: When address is not present in cache:

     We will have to read the data from main memory.

     (i) As address is not present, we will need to check whether any invalid state line is present in the cache. We will do that using find_invalid() command.
        (a) If invalid state cache line is present,
            - We perform bus read operation, so as to get the data in L2 cache.
            - Then we will perform message to cache operation so that L1 can read data from L2 cache.
            - Update PLRU bits.
        (b) If Invalid state cache state is not present, it means we have to evict a cache line. Using Read_PLRU function, we will understand which way was least recently used. Now we need to check state of that particular cache line.
            - If it is in Modified state, we need to get latest data from L1 cache.
            - Then we need to write the data to the main memory using BusOp() function.
            - Then, we need to evict the cache line in the L1 cache as it will be updated.
            - Then, we perform read from DRAM operation, so that we can get the fresh data in L2 cache. Then we will execute the MessageToCache function to send the values to L1 cache, so that required read operation will be performed.
        (c) If it is in Shared or Exclusive state,
            - Then, we need to evict the cache line in the L1 cache as it will be updated.
            - Then, we perform read from DRAM operation, so that we can get the fresh data in L2 cache.
            - Then we will execute the MessageToCache function to send the values to L1 cache, so that required read operation will be performed.

     At the end, we need to update the current state of the updated cache line. So, if there is a HIT or HITM from SnoopResult function, then the current cache line state will be updated as Shared and if there is NOHIT, then current cache line state will be Exclusive.

2) write_request_from_L1_data_cache:

When there is write request from L1 cache

Case A: When address is present in L2 cache:

(iii) Check if that address is present in L2 cache using search_cache() function. When it is present send message to cache to sendline and then getline.
(iv) Update the PLRU bits.
(v) L2 will do busoperation (invalidate), to let other processors and cache know that the line has been modified

Case B: When address is not present in cache:

We will have to read the data from main memory.

(ii) As address is not present, we will need to check whether any invalid state line is present in the cache. We will do that using find_invalid() command.

(d) If invalid state cache line is present,
- We perform bus read operation Read with intent to modify
- Then we will perform message to cache operation to send that line to L1 so that it can perform write operation and then getline
- Update PLRU bits.

(e) If Invalid state cache state is not present, it means we have to evict a cache line. Using Read_PLRU function, we will understand which way was least recently used. Now we need to check state of that particular cache line.
- If it is in Modified state, we need to get latest data from L1 cache.
- Then we need to write the data to the main memory using BusOp() function.
- Then, we need to evict the cache line in the L1 cache as it will be updated.
- Then, we perform busoperation Read with intent to modify
- Then we will execute the MessageToCache function sendline and then getline.

(f) If it is in Shared or Exclusive state,
- Then, we need to evict the cache line in the L1 cache as it will be updated.
- Then, we perform busoperation read with intent to modify
- Then we will execute the MessageToCache function to send the values to L1 cache, so that required write operation will be performed
- Update PLRU bits.

3) Same as operation command 0.

4) Snooped invalidate request:

When we receive snooped invalidate command, we need to check whether that cache line is present in our cache or not. If it is present and it is not in the Invalid state, then we need to update the cache line state to Invalid. Also, we need to tell L1 cache to invalidate that particular cache line.

5) Snooped read request:

When we receive a snooped read request, we understand that another processor is performing read operation. In this case, we need to check whether the same line is present in our cache line or not.

- If it is present and we are in Modified state, then we need to get latest copy of the data from L1 and then write it to main memory. And then we will put the snoop result in which we will indicate that HITM is occurred. And the state will be updated as Shared.
- If we are in Exclusive state, then we will put the snoop result as HIT. And our state will be changed to Shared.
- If we are in Shared state, we will put the snoop result as HIT. And our state will be changed to Shared.
- Otherwise, we will put snoop result as NOHIT.

6)Snooped write request:

When another processor performs the write operation to DRAM, it means it had modified data and should have Invalid command over bus. As we are snooping that Invalid command, we should have changed state of that particular cache line to Invalid. In other words, we should not have line. So, we need not to do any operation when we snoop a write request.

8) Clear the cache and reset all state:

In this each cache line will be cleared and its state will be updated to Invalid.

9) Print contents and state of each valid cache line:
We print the contents and states of each valid cache line in a tabular format.

- Test case:

Set 1659

Hits: 0 Miss: 7

Fill from way 0 to 3, snoop invalidate data at way 2. Fill new data to check it starts filling from way 2. Snoop read request on address which is previously exclusive to check if it changes to shared state.

Set 1328

Hits: 1 Miss: 9

Fill all ways of that set read 2nd way. Have new read/write request to evict LRU. This check PLRU function.

Set 13b

Hits: 8 Miss: 11

Fill all ways of set. Read ways as given 8->2->5->1->7->6->3->4. Therefore, LRU is way 8. Fill new data to check PLRU operation. Given operation 5 to check if nothing happens for that operation.

Set 1197 & 1997

Hit: 0 Miss: 8

Fill all 8 ways. Read any one way and try accessing new data to check PLRU working properly. Perform snoop RWIM to check if that address goes to Invalid state.

Total:

Hits: 9 Miss: 35 Reads: 39 Writes: 5

Tested with few simple test cases to check if proper bus operations are being performed for different states and different command operation.