**MICROPROCESSOR SYSTEM DESIGN**

**ECE 585**

**LLC SIMULATOR**

**Project Report**

Instructor: Professor Mark G. Faust

**Team Members: -**

Sandeep Prabhakaran(sandeep6@pdx.edu)

**TABLE OF CONTENTS**

Introduction 1

Cache Address Mapping 2

Cache Data Structure 3

Design Decisions 4

Testcases For PSUEDO LRU replacement policy 5

Testcases For Write Command

Testcases For Read Command

**Cache Data Structure**

struct {

short int PLRU; ///////////// LSB being the start of the root ////////// {d,e,f,g,b,c,a}//////////////////

int tag\_array [ways]; //////////// Tag array//////// {Messi bits [1:0],tagbits} //////////////

} cache[sets];

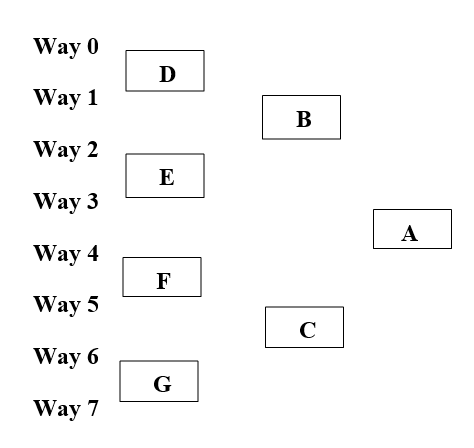
****

Figure PSEUDO LRU replacement structure

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| D | E | F | G | B | C | A |

Figure PSEUDO LRU register per set

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| WAYS | D | E | F | G | B | C | A |
| Way 0 | 1 | X | X | X | 1 | X | 1 |
| Way 1 | 0 | X | X | X | 1 | X | 1 |
| Way 2 | X | 1 | X | X | 0 | X | 1 |
| Way 3 | X | 0 | X | X | 0 | X | 1 |
| Way 4 | X | X | 1 | X | X | 1 | 0 |
| Way 5 | X | X | 0 | X | X | 1 | 0 |
| Way 6 | X | X | X | 1 | X | 0 | 0 |
| Way 7 | X | X | X | 0 | X | 0 | 0 |

Figure Way update table

**Design Decisions**

* Snarfing: If a current address is in modified in the LLC A. If it snooped a Bus Read from LLC B, LLC A does a bus write and go to shared state. LLC B seeing the HITM Generated from LLC A also goes to shared state, snarfing the bus write.
* Bus Read or Bus read exclusive can generate address to DRAM controller which can specify the specific burst order needed. The burst wraps around if the burst is selected from address which is not in multiples of 64.
* This LLC does not send data to L1 in critical word first order or Early restart. It first gets the entire line from the DRAM. Then it sends the entire line after it is been received in order.

**Testcases For PSUEDO LRU replacement policy**

|  |  |
| --- | --- |
| TESTCASE | DESCRIPTION |
| 1 | Initial Replacement  Method: First we are doing a read to different lines mapping to a set 8 times. Then we are doing a read to a different line mapping to the same set and observe the results. Way 0 should be replaced. |
| 2 | Filling the line and reading way 0.  Method: First we are doing a read to different lines mapping to a set 8 times. Then we are doing a read to a line in way 0. Then we are doing a read to a different line mapping to the same set for replacing a line and observe the results. Way 4 should be replaced. |
| 3 | Filling the line and reading way 0, way 4.  Method: First we are doing a read to different lines mapping to a set 8 times. Then we are doing reads to the lines in this order way 0, way 4. Then we are doing a read to a different line mapping to the same set for replacing a line and observe the results. Way 2 should be replaced. |
| 4 | Filling the line and reading way 0, way 4, way 2.  Method: First we are doing a read to different lines mapping to a set 8 times. Then we are doing reads to the lines in this order way 0, way 4, way 2. Then we are doing a read to a different line mapping to the same set for replacing a line and observe the results. Way 6 should be replaced. |
| 5 | Filling the line and reading way 0, way 4, way 2, way 6.  Method: First we are doing a read to different lines mapping to a set 8 times. Then we are doing reads to the lines in this order way 0, way 4, way 2, way 6. Then we are doing a read to a different line mapping to the same set for replacing a line and observe the results. Way 1 should be replaced. |
| 6 | Filling the line and reading way 0, way 4, way 2, way 6, way 1.  Method: First we are doing a read to different lines mapping to a set 8 times. Then we are doing reads to the lines in this order way 0, way 4, way 2, way 6, way 1. Then we are doing a read to a different line mapping to the same set for replacing a line and observe the results. Way 5 should be replaced. |
| 7 | Filling the line and reading way 0, way 4, way 2, way 6, way 1, way 5.  Method: First we are doing a read to different lines mapping to a set 8 times. Then we are doing reads to the lines in this order way 0, way 4, way 2, way 6, way 1, way 5. Then we are doing a read to a different line mapping to the same set for replacing a line and observe the results. Way 3 should be replaced. |
| 8 | Filling the line and reading way 0, way 4, way 2, way 6, way 1, way 5, way 3.  Method: First we are doing a read to different lines mapping to a set 8 times. Then we are doing reads to the lines in this order way 0, way 4, way 2, way 6, way 1, way 5, way 3. Then we are doing a read to a different line mapping to the same set for replacing a line and observe the results. Way 7 should be replaced. |
| 9 | Filling the line and reading way 0, way 4, way 2, way 6, way 1, way 5, way 3, way 7.  Method: First we are doing a read to different lines mapping to a set 8 times. Then we are doing reads to the lines in this order way 0, way 4, way 2, way 6, way 1, way 5, way 3, way 7. Then we are doing a read to a different line mapping to the same set for replacing a line and observe the results. Way 0 should be replaced. |

**Testcases For Write Command**

|  |  |
| --- | --- |
| TESTCASE | DESCRIPTION |
| 10 | The current state of a line is in shared. Doing a write to the same line to check if it is a cache hit.  Method: First, we are giving appropriate commands to get the line to shared state, then we are doing a write to the same line to observe the results. |
| 11 | The current state of a line is invalid. Doing a write to the same line to check if it is a cache miss.  Method: First, we are giving appropriate commands to get a single line to invalid state, then we are doing a write to the same line to observe the results. |
| 12 | The current state of a line is modified. Doing a write to the same line to check if it is a cache hit.  Method: First, we are giving appropriate commands to get a single line to modified state, then we are doing a write to the same line to observe the results. |
| 13 | The current state of a line is exclusive. Doing a write to the same line to check if it is a cache hit.  Method: First, we are giving appropriate commands to get a single line to exclusive state, then we are doing a write to the same line to observe the results. |
| 14 | Cache miss when doing a write. Empty way found.  Method: We are reading different lines mapping to a set 8 times. And invalidating random lines through appropriate commands. Then writing different sets of lines to the same set to see if we are getting the lines placed in the empty ways. |
| 15 | Cache miss when doing a write. Conflict miss. Line to be replaced is modified.  Method: We are writing different lines mapping to a set 8 times. Then we are writing a different line mapping to the same set to observe the results. |
| 16 | Cache miss when doing a write. Conflict miss. Line to be replaced is shared.  Method: We are reading different lines mapping to a set 8 times. Then we are giving appropriate commands to get a single line to shared at way 0. Then we are doing a write to specifically replace the shared line and observe the results. |
| 17 | Cache miss when doing a write. Conflict miss. Line to be replaced is exclusive.  Method: We are reading different lines mapping to a set 8 times, also making sure that the line to be replaced is exclusive at way 0. Then we are doing a write to specifically replace the exclusive line and observe the results. |

**Testcases For READ Command**

|  |  |
| --- | --- |
| TESTCASE | DESCRIPTION |
| trace\_18.txt | The current state of a line is in shared. Doing a read to the same line to check if it is a cache hit.  Method: First, we are giving appropriate commands to get the line to shared state, then we are doing a read to the same line to observe the results. |
| trace\_19.txt | The current state of a line is invalid. Doing a read to the same line to check if it is a cache miss.  Method: First, we are giving appropriate commands to get a single line to invalid state, then we are doing a read to the same line to observe the results. |
| trace\_20.txt | The current state of a line is modified. Doing a read to the same line to check if it is a cache hit.  Method: First, we are giving appropriate commands to get a single line to modified state, then we are doing a read to the same line to observe the results. |
| trace\_21.txt | The current state of a line is exclusive. Doing a read to the same line to check if it is a cache hit.  Method: First, we are giving appropriate commands to get a single line to exclusive state, then we are doing a read to the same line to observe the results. |
| trace\_22.txt | Cache miss when doing a read. Empty way found.  Method: We are reading different lines mapping to a set 8 times. And invalidating a random line through appropriate commands. Then reading a different lines to the same set to see if we are getting the lines placed in the empty ways. |
| trace\_23.txt | Cache miss when doing a read. Conflict miss. Line to be replaced is modified.  Method: We are reading different lines mapping to a set 8 times. Then we are reading a different line mapping to the same set to observe the results. |
| trace\_24.txt | Cache miss when doing a read. Conflict miss. Line to be replaced is shared.  Method: We are reading different lines mapping to a set 8 times. Then we are giving appropriate commands to get a shared single line. Then we are doing a read to specifically replace the shared line and observe the results. |
| trace\_25.txt | Cache miss when doing a read. Conflict miss. Line to be replaced is shared.  Method: We are reading different lines mapping to a set 8 times, also making sure that the line to be replaced is shared at way 0. Then we are doing a read to specifically replace the shared line with a snoop result HIT line and observe the results. |
| trace\_26.txt | Cache miss when doing a read. Conflict miss. Line to be replaced is shared.  Method: We are reading different lines mapping to a set 8 times, also making sure that the line to be replaced is shared at way 0. Then we are doing a read to specifically replace the shared line with a snoop result HITM line and observe the results. |
| trace\_27.txt | Cache miss when doing a read. Conflict miss. Line to be replaced is shared.  Method: We are reading different lines mapping to a set 8 times, also making sure that the line to be replaced is shared at way 0. Then we are doing a read to specifically replace the shared line with a snoop result NOHIT line and observe the results. |
| trace\_28.txt | Cache miss when doing a read. Conflict miss. Line to be replaced is Exclusive.  Method: We are reading different lines mapping to a set 8 times, also making sure that the line to be replaced is shared at way 0. Then we are doing a read to specifically replace the shared line with a snoop result HIT line and observe the results. |
| trace\_29.txt | Cache miss when doing a read. Conflict miss. Line to be replaced is Exclusive.  Method: We are reading different lines mapping to a set 8 times, also making sure that the line to be replaced is shared at way 0. Then we are doing a read to specifically replace the shared line with a snoop result HITM line and observe the results. |
| trace\_30.txt | Cache miss when doing a read. Conflict miss. Line to be replaced is Exclusive.  Method: We are reading different lines mapping to a set 8 times, also making sure that the line to be replaced is shared at way 0. Then we are doing a read to specifically replace the shared line with a snoop result NOHIT line and observe the results. |
| trace\_31.txt | Cache miss when doing a read. Conflict miss. Line to be replaced is Modified.  Method: We are reading different lines mapping to a set 8 times, also making sure that the line to be replaced is shared at way 0. Then we are doing a read to specifically replace the shared line with a snoop result HIT line and observe the results. |
| trace\_32.txt | Cache miss when doing a read. Conflict miss. Line to be replaced is Modified.  Method: We are reading different lines mapping to a set 8 times, also making sure that the line to be replaced is shared at way 0. Then we are doing a read to specifically replace the shared line with a snoop result HITM line and observe the results. |
| trace\_33.txt | Cache miss when doing a read. Conflict miss. Line to be replaced is Modified.  Method: We are reading different lines mapping to a set 8 times, also making sure that the line to be replaced is shared at way 0. Then we are doing a read to specifically replace the shared line with a snoop result NOHIT line and observe the results. |

**Testcases for Snooping command**

|  |  |
| --- | --- |
| 34 | Snooping BusRead while the cache state being exclusive  Method: Bring the cache in exclusive state by issuing a READ with hit/hitm = 0. Then snoop a BusRead and test whether the cache is going in shared state or not. |
| 35 | Snooping BusRead while the cache state being modified.  Method: Bring the cache in modified state by issuing a WRITE. Then snoop a BusRead and test whether the cache is going in shared state or not and whether it gets line from L1, invalidates L1 issues a BusWrite. |
| 36 | Snooping BusRead while the cache state being shared  Method: Bring the cache in shared state by issuing a READ with hit/hitm = 1. Then snoop a BusRead and test whether the cache is staying in the shared state or not. |
| 37 | Snooping invalid while the cache state being shared  Method Bring the cache in shared state by issuing a READ with hit/hitm = 1. Then snoop a invalid and test whether the cache is getting in invalid state or not and it invalidates l1 as well |
| 38 | Snooping RWIM while the cache state being shared  Method Bring the cache in shared state by issuing a READ with hit/hitm = 1. Then snoop a invalid and test whether the cache is getting in invalid state or not and it invalidates l1 as well |
| 39 | Snooping RWIM while the cache state being modified  Method: Bring the cache in modified state by issuing a WRITE. Then snoop a RWIM and test whether the cache is going in invalid state or not and whether it gets line from L1, invalidates L1 issues a BusWrite. |
| 40 | Snooping RWIM while the cache state being exclusive  Method: Bring the cache in exclusive state by issuing a READ with hit/hitm = 0. Then snoop a RWIM and test whether the cache is going in invalid state or not and it invalidates l1 as well |