Project

Report on

Last Level Cache Design and Simulation

by

Aakash
Siddharth

Rajani Kallur

Satyajit Deokar

Siddesh Patil

Under the Guidance of

Prof. Mark G. Faust

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING
PORTLAND STATE UNIVERSITY

1. Introduction:

The demand for faster computer performance grows as technology advances. To attain increased performance, processors are designed to operate at frequencies ranging from thousands of Megahertz to tens of gigahertz. Even though the processor operates at such high frequencies, memory systems cannot keep up, necessitating the hunt for an alternative solution. The problem was handled utilizing the "caching" approach. Before examining all of memory, processor requests are directed through a tiny memory unit known as cache. Cache holds critical information, such as data or instructions, that are regularly needed by processors, allowing processor demands to be performed as rapidly as feasible. If the information is not already in cache.

2. Applications:

The concept of caching can be applied in various computing domains such as:

- **Browser Caching**: Stores website data locally for faster loading and reduced network usage.
- **CPU Instruction Cache**: Speeds up instruction access by storing recently fetched instructions.
- Caches in the memory hierarchy, serving as lower-level or higher-level caches.
- Translation Look Aside Buffers (TLB) in Virtual memory.

3. Technical Specification:

The last level cache (L2) design implemented in the project has the following specifications:

- Total Capacity = 16MB
- Byte Line = 64-byte lines
- Associativity = 16-way set
- Policy decisions for write miss = Write allocate
- Coherence Protocol = MESI
- Replacement Policy = pseudo-LRU scheme

L1 cache specifications:

- Byte Line = 64-byte lines
- Associativity = 4-way set
- The first write to a line is write-once policy: the first write to a line is write through;
- Subsequent writes to the same line are write-back
- Must maintain inclusivity

4. Assumptions:

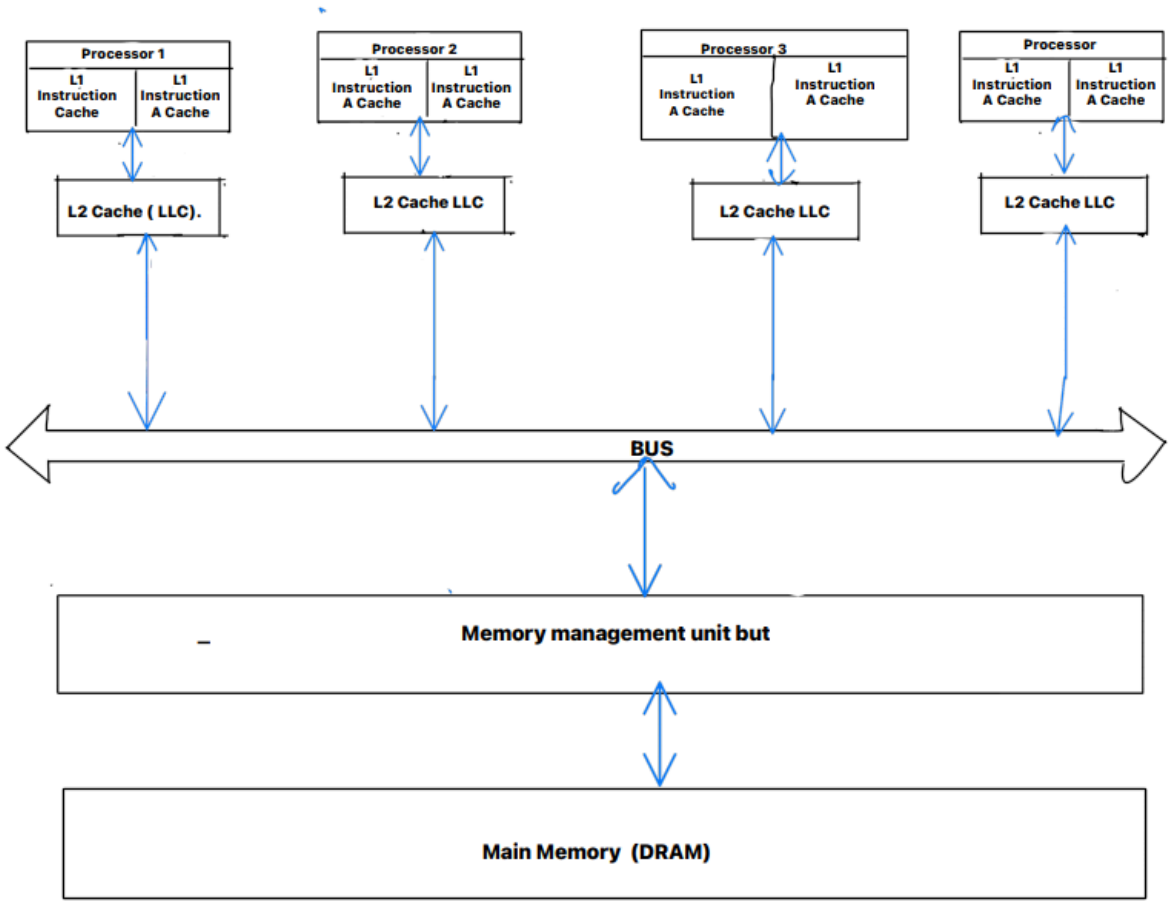
We've made the following assumptions in designing the L1:

- Last 2-digits of the address determines HIT/HITM/MISS
- henever L1 receives EVICTLINE it will send the line to L2 and will invalidate its copy.
- Whenever L2 has data in state (M), then we consider its presence in L1 cache, so we do GETLINE and INVALIDATELINE. If snooped operation then we do EVICTLINE for processorside operation

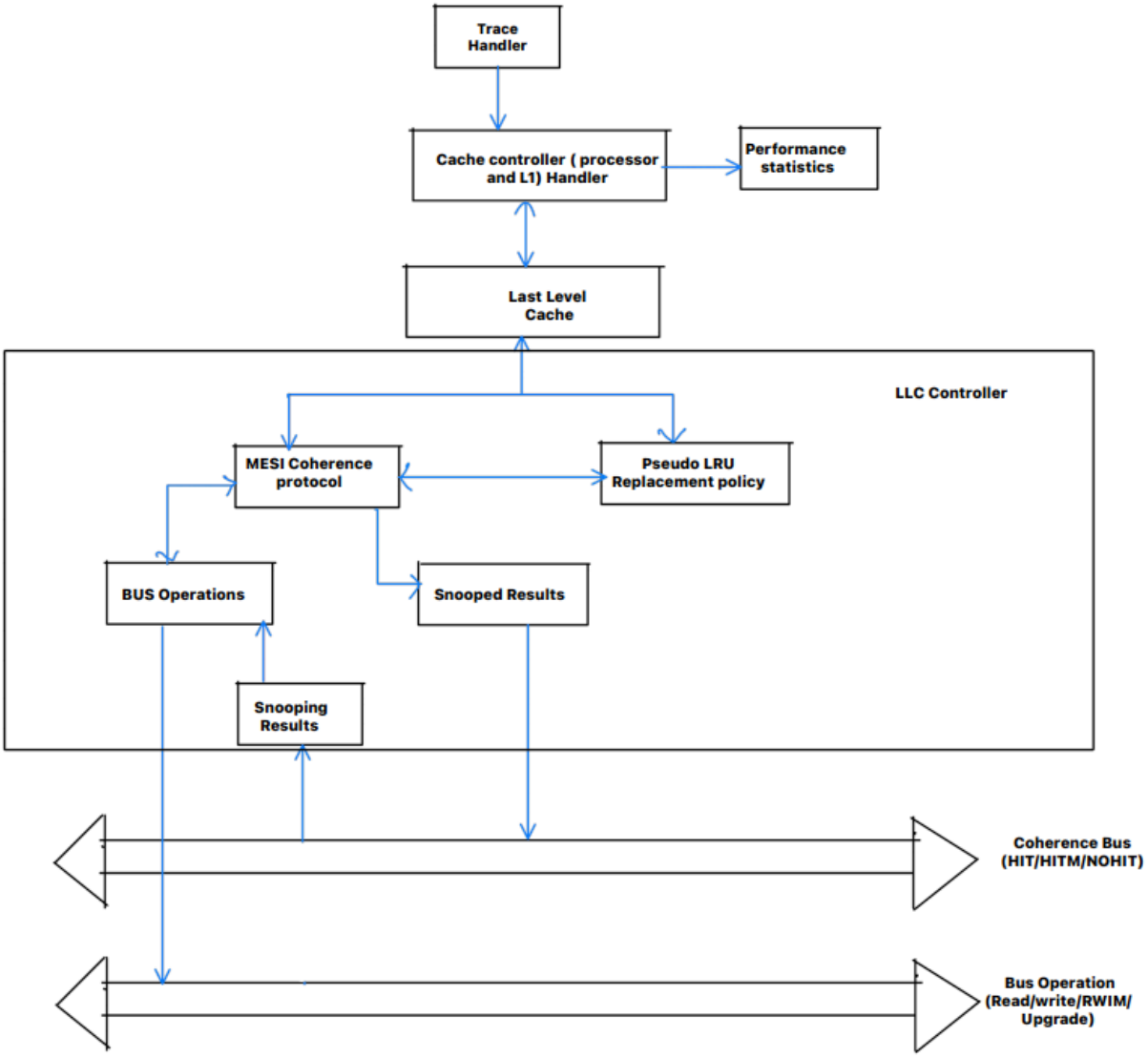
5. Modes Used:

- **Silent mode:** The simulation outputs only the usage statistics summary and the response to 9's in the trace file.
- **Normal mode:** The simulation outputs all details, including bus operations, snoop results, messages from L2 to L1 cache, and first mode.
- **Debug mode:** to display debug information during the simulation using Conditional compilation.

6. Shared memory configuration:



7. Design Architecture:



8. Source Code:

a. SystemVerilog Packages:

All parameters and functions used in the design are defined in the following package files.

pkg_bus.sv, pkg_plru.sv, pkg_line.sv

b. CacheDesign:

Implements the Last Level Cache as per the project specification.

TOP.sv

9. Testing Strategies:

a. Basic Ports & Parameters test

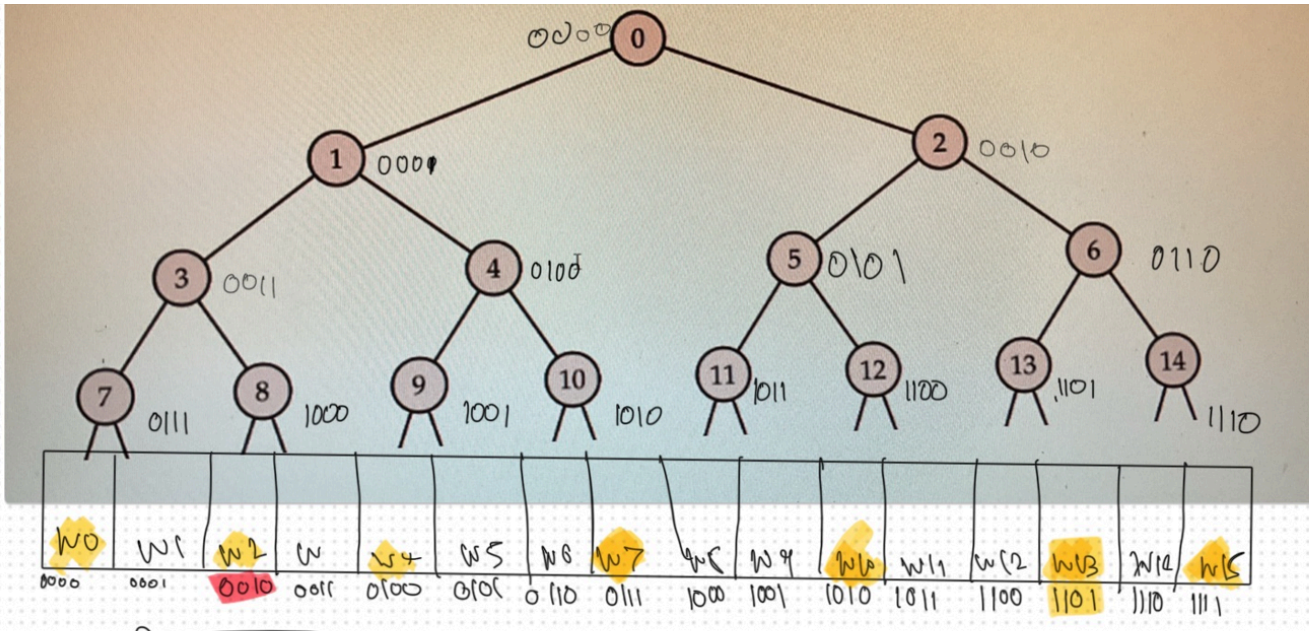
- Tests the counters (Hit and miss), parameters and **enums** used and ports

b. Testing basic reads and writes to the cache

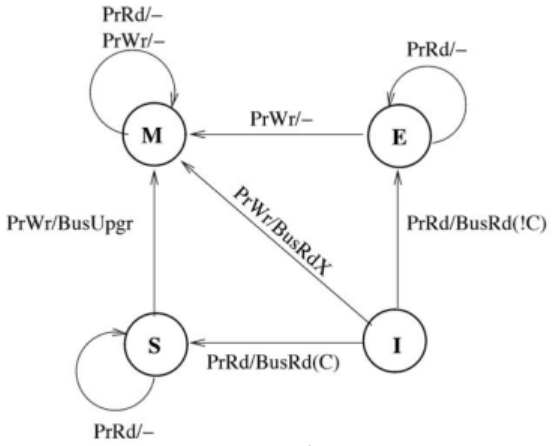
- Write to an address in the L2 cache (with an empty cache set), followed by sequential reads from the same location, resulting in read hits.
- Write to all ways of an empty cache set, then read from all of them in succession, resulting in a read hit.

c. Testing the PLRU

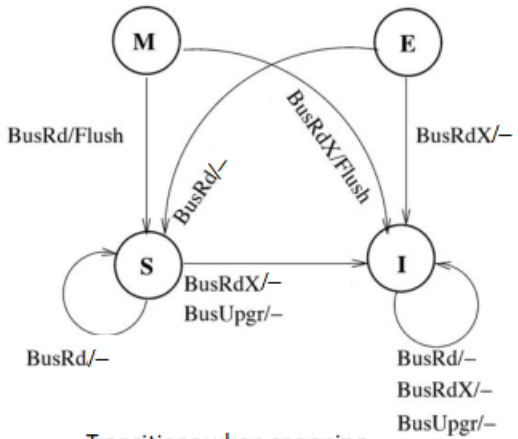
- Complete a set of 16 writes, during eviction call the Victim way function which will provide the PLRU way.



0. Test Cases for MESI FSM



Transitions when CPU Request



Transitions when snooping

Case analysis 1

Case n= 0 read request from L1 data cache

a) Invalid (I):
- Bus operation: READ
- Snoop results:
  HIT: State change (I) □ (E)
  NOHIT: state change (I) □(S)
- L2 – L1 message type: SENDLINE

b) Modified (M)
- L2 – L1 message type: SENDLINE
- state change: remain in same state(M)

c) Exclusive (E)
- L2 – L1 message type: SENDLINE
- state change: remain in same state(E)

d) Shared (S)
- L2 – L1 message type: SENDLINE
- state change: remain in same state(S)

Case n=1 write request from L1 data cache

a) Invalid (I):
- Bus operation: RWIM
- State change (I) □ (M)
- L2 – L1 message type: SENDLINE

b) Modified (M)
- state change: remain in same state(M)

c) Exclusive (E)

- state change: remain in same state(M)

d) Shared (S)
- L2 – L1 message type: SENDLINE
- state change: jump to (M)

Case n=2 read request from L1 instruction cache

a) Invalid (I):
- Bus operation: READ
- Snoop results:
  HIT: State change (I) □ (S)
  NOHIT: state change (I) □(E)
- L2 – L1 message type: SENDLINE

b) Modified (M): not in picture

c) Exclusive (E)
- L2 – L1 message type: SENDLINE
- state change: remain in same state(E)

d) Shared (S)
- L2 – L1 message type: SENDLINE
- state change: remain in same state(S)

Case n=3 snooped read request //(BusRd)

a) Invalid (I):
- state change: remain in same state(I)

b) Modified (M):
- Snoop results:
  HITM: State change (M) □ (S)

c) Exclusive (E)
- Snoop results:
  HIT: State change (E) □ (S)

d) Shared (S)
- Snoop results:
  HIT: remain in same state(S)

Case n=4 snooped write request //(BusUpgr)

a) Invalid (I):
- state change: remain in same state(I)

b) Modified (M): not in picture

c) Exclusive (E): not in picture

d) Shared (S)
- Snoop results:
  HIT: State change (S) □ (I)
- L2 – L1 message type: INVALIDATELINE

Case n=5 snooped read with intent to modify request //(BusRdX)

a) Invalid (I):
   - state change: remain in same state(I)
b) Modified (M):
   - Bus operation: WRITE
   - Snoop results:
      HITM: State change (M) □ (I)
   - L2 – L1 message type: INVALIDATELINE
c) Exclusive (E)
   - Snoop results:
      HIT: State change (E) □ (I)
   - L2 – L1 message type: INVALIDATELINE
d) Shared (S)
   - Snoop results:
      HIT: State change (S) □ (I)
   - L2 – L1 message type: INVALIDATELINE

Case n=6 snooped invalidate command //(Busupgrade)

a) Invalid (I): not in picture
b) Modified (M): not in picture
c) Exclusive (E): not in picture
d) Shared (S)
   State change (S) □ (I)

Case n=8 clear the cache and reset all state

- clear the cache and reset all the state
- Reset cache to zero
- change state to Invalid

Case n=9 print contents and state of each valid cache line (doesn't end simulation!)

- Print contents and state of each valid cache line

Case analysis 1.1

Case n= 0 read request from L1 data cache

If cache HIT:
   -L2-L1 message type: SENDLINE
If cache MISS: Eviction
   a) Modified (M)
      - L2 – L1 message type:EVICTLINE (processor side)
      - Bus operation : WRITE (evicted line)
        READ (new line)
      - state change: remain in same state(M)
   b) Invalid (I):
      - Bus operation: READ

- Snoop results:
  HIT/HITM: State change (I) □ (S)
  NOHIT: state change (I) □(E)

c) Shared (S)
   - state change: remain in same state(S)

d) Exclusive (E)
   - state change: remain in same state(E)

Case n=1 write request from L1 data cache

If cache HIT:
Modified (M) / Shared(S) / Exclusive (E) : remains in the same states
If cache MISS: Eviction

a) Modified (M)
   - Bus operation: WRITE(evicted line)
   -L2-L1 message type: EVICTLINE
   - Bus operation: RWIM (new line)
   -L2-L1 message type: SENDLINE (evicted)
   - State change: remains in the same state (M)

b) Invalid (I):
   - Bus operation: RWIM
   - State change (I) □ (M)
   - L2 – L1 message type: SENDLINE (new line)

c) Exclusive (E)
   - Bus operation: RWIM
   - L2 – L1 message type: EVICTLINE
     INVALIDATELINE(evicted line)
   - state change: jump (E) □(M)

d) Shared (S)
   - Bus operation: RWIM
   - L2 – L1 message type: EVICTLINE
     INVALIDATELINE(evicted line)
   - state change: jump to (M)

Case n=2 read request from L1 instruction cache

If cache HIT:

- L2-L1 message type: SENDLINE (evicted)

If cache MISS: Eviction

a) Invalid (I):
   - Bus operation: READ (newline)

- State change: remain in the same state (I)
- L2 – L1 message type: EVICTLINE
- Snoop results:
  HIT: State change (I) □ (S)
  NOHIT: state change (I) □(E)

b) Exclusive (E)
- Bus operation: READ (newline)
- L2 – L1 message type: SENDLINE
- state change: jump (E) □(I)
- Snoop results:
  HIT: State change (E) □ (S)
  NOHIT: state change (E) □(E)

c) Shared (S)
- Bus operation: READ (newline)
- L2 – L1 message type: SENDLINE
- state change: jump (S) □(I)
- Snoop results:
  HIT: State change (I) □ (S)
  NOHIT: state change (I) □(E)

Case n=3 snooped read request //(BusRd)

If cache HITM:

Modified (M):
- Bus operation: WRITE
- State change: jump (M) → (S)
- L2 – L1 message type: GETLINE(newline)

If cache HIT: Eviction

Exclusive (E)
- Bus operation: NO operation
- state change: jump (E) □(S)

Shared (S)
- Bus operation: NO operation
- state change: remains in the same state (S)

If cache NOHIT: Invalidate(I)

Case n=4 snooped write request //(BusUpgr)

If cache HITM: BUG

If cache HIT: Eviction

Exclusive (E)

- Bus operation: NO operation
- L2 – L1 message type: INVALIDATELINE
- state change: jump (E) □(I)

Shared (S)

- Bus operation: NO operation
- L2 – L1 message type: INVALIDATELINE
- state change: jump (S) □(I)

Case n=5 snooped read with intent to modify request //(BusRdX)

If cache HITM:

a) Exclusive (M)
   - Bus operation: WRITE
   - L2 – L1 message type: GETLINE (new line)
     INVALIDATELINE (evictedline)
   - state change: jump (M) □(I)

If cache HIT: Eviction

b) Exclusive (E)
   - Bus operation: NO operation
   - L2 – L1 message type: INVALIDATELINE
   - state change: jump (E) □(I)

c) Shared (S)
   - Bus operation: NO operation
   - L2 – L1 message type: INVALIDATELINE
   - state change: jump (S) □(I)

Case n=6 snooped invalidate command //(Busupgrade)

If cache HITM:

a) Exclusive (M)
   - Bus operation: WRITE
   - L2 – L1 message type: GETLINE (new line)

INVALIDATELINE (evictedline)
- state change: jump (M) □(I)

If cache HIT: Eviction

b) Exclusive (E)
- Bus operation: NO operation
- L2 – L1 message type: INVALIDATELINE
- state change: jump (E) □(I)

c) Shared (S)
- Bus operation: NO operation
- L2 – L1 message type: INVALIDATELINE
- state change: jump (S) □(I)

Case n=8 clear the cache and reset all state
- clear the cache and reset all the state
- Reset cache to zero
- change state to Invalid

Case n=9 print contents and state of each valid cache line (doesn't end simulation!)
- Print contents and state of each valid cache line

Output:

Example: SILENT Mode Transcript sample file

```
vsim -c LLC_Cache -do "run -all" +trace_file=traces/t17.din
# End time: 22:52:12 on Dec 10,2024, Elapsed time: 0:19:08
# Errors: 0, Warnings: 1
# vsim -c LLC_Cache -do "run -all" "+trace_file=traces/t17.din"
# Start time: 22:52:12 on Dec 10,2024
# ** Note: (vsim-3813) Design is being optimized due to module recompilation...
# Loading sv_std.std
# Loading work.pkg_line(fast)
# Loading work.pkg_bus(fast)
# Loading work.pkg_plru(fast)
# Loading work.TOP_sv_unit(fast)
# Loading work.LLC_Cache(fast)
# run -all
#
# -> Read request from L1 data cache, Address: 00000002
```

```
# BusOp: READ, Address: 00000002, Snoop Result: NOHIT
# L2: SENDLINE 00000000
#
# -> Read request from L1 data cache, Address: 00200000
# BusOp: READ, Address: 00200000, Snoop Result: HIT
# L2: SENDLINE 00200000
#
# -> Write request from L1 data cache, Address: 00400002
# BusOp: RWIM, Address: 00400002, Snoop Result: NOHIT
# L2: SENDLINE 00000000
#
# -> Read request from L1 data cache, Address: 00600002
# BusOp: READ, Address: 00600002, Snoop Result: NOHIT
# L2: SENDLINE 00600000
#
# -> Read request from L1 data cache, Address: 00800000
# BusOp: READ, Address: 00800000, Snoop Result: HIT
# L2: SENDLINE 00800000
#
# -> Write request from L1 data cache, Address: 00a00000
# BusOp: RWIM, Address: 00a00000, Snoop Result: HIT
# L2: SENDLINE 00000000
#
# -> Printing contents and state of each valid cache line.
# Set: 0, Way: 0, MESI: E, Tag: 000
# Set: 0, Way: 1, MESI: S, Tag: 002
# Set: 0, Way: 2, MESI: M, Tag: 004
# Set: 0, Way: 3, MESI: E, Tag: 006
# Set: 0, Way: 4, MESI: S, Tag: 008
# Set: 0, Way: 5, MESI: M, Tag: 00a
#
# -> Read request from L1 data cache, Address: 00000000
# L2: SENDLINE 00000000
#
# -> Read request from L1 data cache, Address: 00200000
# L2: SENDLINE 00200000
#
# -> Read request from L1 data cache, Address: 00400000
# L2: SENDLINE 00400000
#
# -> Write request from L1 data cache, Address: 00600000
# BusOp: INVALIDATE, Address: 00600000, Snoop Result: HIT
#
# -> Write request from L1 data cache, Address: 00800000
# BusOp: INVALIDATE, Address: 00800000, Snoop Result: HIT
#
# -> Write request from L1 data cache, Address: 00a00000
```

```
#
# -> Printing contents and state of each valid cache line.
# Set: 0, Way: 0, MESI: E, Tag: 000
# Set: 0, Way: 1, MESI: S, Tag: 002
# Set: 0, Way: 2, MESI: M, Tag: 004
# Set: 0, Way: 3, MESI: M, Tag: 006
# Set: 0, Way: 4, MESI: M, Tag: 008
# Set: 0, Way: 5, MESI: M, Tag: 00a
#
# No. of cache hits = 6
# No. of cache misses = 6
# No. of cache writes = 5
# No. of cache reads = 7
#
# ====================================
# ANKARA MESSI ANKARA MESSI MESSI MESSI ANKARA MESSI GOAAAAAAAAAAAAAAL
====================================!
# Cache Hit Ratio = 0.500000
```

Example: NORMAL Mode Transcript sample file

```
vlog +define+DEBUG TOP.sv
# QuestaSim-64 vlog 2021.3_1 Compiler 2021.08 Aug 15 2021
# Start time: 22:29:15 on Dec 10,2024
# vlog -reportprogress 300 "+define+DEBUG" TOP.sv
# -- Compiling package pkg_line
# -- Compiling package pkg_line
# ** Warning: ** while parsing file included at TOP.sv(3)
# ** while parsing file included at pkg_bus.sv(1)
# ** at pkg_line.sv(1): (vlog-2275) Existing package 'pkg_line' at line 1 will be
overwritten.
# -- Compiling package pkg_bus
# -- Importing package pkg_line
# -- Compiling package pkg_line
# ** Warning: ** while parsing file included at TOP.sv(4)
# ** while parsing file included at pkg_plru.sv(1)
# ** at pkg_line.sv(1): (vlog-2275) Existing package 'pkg_line' at line 1 will be
overwritten.
# -- Compiling package pkg_plru
# ** Warning: TOP.sv(6): (vlog-13233) Design unit "TOP_sv_unit" already exists and
will be overwritten. Design unit compiled with different set of options.
# -- Compiling package TOP_sv_unit
# -- Importing package pkg_plru
# -- Importing package pkg_bus
# -- Compiling module LLC_Cache
```

```
#
# Top level modules:
#      LLC_Cache
# End time: 22:29:15 on Dec 10,2024, Elapsed time: 0:00:00
# Errors: 0, Warnings: 3
vsim -c LLC_Cache -do "run -all" +trace_file=traces/t0.din
# vsim -c LLC_Cache -do "run -all" "+trace_file=traces/t0.din"
# Start time: 22:32:37 on Dec 10,2024
# ** Note: (vsim-3812) Design is being optimized...
# Loading sv_std.std
# Loading work.pkg_line(fast)
# Loading work.pkg_bus(fast)
# Loading work.pkg_plru(fast)
# Loading work.TOP_sv_unit(fast)
# Loading work.LLC_Cache(fast)
# run -all
# A working code snippet to read and parse an input trace file, using a default
file if the user does not specify a filename.
# Using input trace file: traces/t0.din
# File opened successfully: traces/t0.din
#
################ Initializing Cache ################
# Parsed: n = 0,
# Address: Tag[31:20] = 000, Index[19:6] = 0000, Offset[5:0] = 00
#
# -> Read request from L1 data cache, Address: 00000000
# Cache miss for address 00000000
# Found Invalid MESI state in Way: 0
# The victim found in way 0, and state I
# BusOp: READ, Address: 00000000, Snoop Result: HIT
# Updating for Way:           0
# PLRU BitUpdated PLRU[          0] = 0
# PLRU BitUpdated PLRU[          1] = 0
# PLRU BitUpdated PLRU[          3] = 0
# PLRU BitUpdated PLRU[          7] = 0
# Snoop result is : HIT
# Updated MESI state is S
# L2: SENDLINE 00000000
# Parsed: n = 0,
# Address: Tag[31:20] = 000, Index[19:6] = 0000, Offset[5:0] = 3f
#
# -> Read request from L1 data cache, Address: 0000003f
# Cache hit for address 3f | In way 0
# Updating for Way:           0
# PLRU BitUpdated PLRU[          0] = 0
# PLRU BitUpdated PLRU[          1] = 0
# PLRU BitUpdated PLRU[          3] = 0
```

```
# PLRU BitUpdated PLRU[          7] = 0
# L2: SENDLINE 0000003f
# Parsed: n = 9,
# Address: Tag[31:20] = 000, Index[19:6] = 0000, Offset[5:0] = 00
#
# -> Printing contents and state of each valid cache line.
# Set: 0, Way: 0, MESI: S, Tag: 000
#
# No. of cache hits = 1
# No. of cache misses = 1
# No. of cache writes = 0
# No. of cache reads = 2
#
# Finished reading from traces/t0.din.
# ====================================
# ANKARA MESSI ANKARA MESSI MESSI MESSI ANKARA MESSI GOAAAAAAAAAAAAAAL
====================================!
# Cache Hit Ratio = 0.500000
vsim -c LLC_Cache -do "run -all" +trace_file=traces/t17.din
# End time: 22:33:04 on Dec 10,2024, Elapsed time: 0:00:27
# Errors: 0, Warnings: 5
# vsim -c LLC_Cache -do "run -all" "+trace_file=traces/t17.din"
# Start time: 22:33:04 on Dec 10,2024
# ** Note: (vsim-8009) Loading existing optimized design _opt1
# Loading sv_std.std
# Loading work.pkg_line(fast)
# Loading work.pkg_bus(fast)
# Loading work.pkg_plru(fast)
# Loading work.TOP_sv_unit(fast)
# Loading work.LLC_Cache(fast)
# run -all
# A working code snippet to read and parse an input trace file, using a default
file if the user does not specify a filename.
# Using input trace file: traces/t17.din
# File opened successfully: traces/t17.din
#
############### Initializing Cache ###############
# Parsed: n = 0,
# Address: Tag[31:20] = 000, Index[19:6] = 0000, Offset[5:0] = 02
#
# -> Read request from L1 data cache, Address: 00000002
# Cache miss for address 00000002
# Found Invalid MESI state in Way: 0
# The victim found in way 0, and state I
# BusOp: READ, Address: 00000002, Snoop Result: NOHIT
# Updating for Way:           0
# PLRU BitUpdated PLRU[          0] = 0
```

```
# PLRU BitUpdated PLRU[          1] = 0
# PLRU BitUpdated PLRU[          3] = 0
# PLRU BitUpdated PLRU[          7] = 0
# Snoop result is : NOHIT
# Updated MESI state is E
# L2: SENDLINE 00000000
# Parsed: n = 0,
# Address: Tag[31:20] = 002, Index[19:6] = 0000, Offset[5:0] = 00
#
# -> Read request from L1 data cache, Address: 00200000
# Cache miss for address 00200000
# Found Invalid MESI state in Way: 1
# The victim found in way 1, and state I
# BusOp: READ, Address: 00200000, Snoop Result: HIT
# Updating for Way:           1
# PLRU BitUpdated PLRU[          0] = 0
# PLRU BitUpdated PLRU[          1] = 0
# PLRU BitUpdated PLRU[          3] = 0
# PLRU BitUpdated PLRU[          7] = 1
# Snoop result is : HIT
# Updated MESI state is S
# L2: SENDLINE 00200000
# Parsed: n = 1,
# Address: Tag[31:20] = 004, Index[19:6] = 0000, Offset[5:0] = 02
#
# -> Write request from L1 data cache, Address: 00400002
# Cache miss for address 00400002
# Found Invalid MESI state in Way: 2
# The victim found in way 2, and state I
# Updating for Way:           2
# PLRU BitUpdated PLRU[          0] = 0
# PLRU BitUpdated PLRU[          1] = 0
# PLRU BitUpdated PLRU[          3] = 1
# PLRU BitUpdated PLRU[          8] = 0
# BusOp: RWIM, Address: 00400002, Snoop Result: NOHIT
# L2: SENDLINE 00000000
# Updated MESI state is M
# Parsed: n = 0,
# Address: Tag[31:20] = 006, Index[19:6] = 0000, Offset[5:0] = 02
#
# -> Read request from L1 data cache, Address: 00600002
# Cache miss for address 00600002
# Found Invalid MESI state in Way: 3
# The victim found in way 3, and state I
# BusOp: READ, Address: 00600002, Snoop Result: NOHIT
# Updating for Way:           3
# PLRU BitUpdated PLRU[          0] = 0
```

```
# PLRU BitUpdated PLRU[          1] = 0
# PLRU BitUpdated PLRU[          3] = 1
# PLRU BitUpdated PLRU[          8] = 1
# Snoop result is : NOHIT
# Updated MESI state is E
# L2: SENDLINE 00600000
# Parsed: n = 0,
# Address: Tag[31:20] = 008, Index[19:6] = 0000, Offset[5:0] = 00
#
# -> Read request from L1 data cache, Address: 00800000
# Cache miss for address 00800000
# Found Invalid MESI state in Way: 4
# The victim found in way 4, and state I
# BusOp: READ, Address: 00800000, Snoop Result: HIT
# Updating for Way:           4
# PLRU BitUpdated PLRU[          0] = 0
# PLRU BitUpdated PLRU[          1] = 1
# PLRU BitUpdated PLRU[          4] = 0
# PLRU BitUpdated PLRU[          9] = 0
# Snoop result is : HIT
# Updated MESI state is S
# L2: SENDLINE 00800000
# Parsed: n = 1,
# Address: Tag[31:20] = 00a, Index[19:6] = 0000, Offset[5:0] = 00
#
# -> Write request from L1 data cache, Address: 00a00000
# Cache miss for address 00a00000
# Found Invalid MESI state in Way: 5
# The victim found in way 5, and state I
# Updating for Way:           5
# PLRU BitUpdated PLRU[          0] = 0
# PLRU BitUpdated PLRU[          1] = 1
# PLRU BitUpdated PLRU[          4] = 0
# PLRU BitUpdated PLRU[          9] = 1
# BusOp: RWIM, Address: 00a00000, Snoop Result: HIT
# L2: SENDLINE 00000000
# Updated MESI state is M
# Parsed: n = 9,
# Address: Tag[31:20] = 000, Index[19:6] = 0000, Offset[5:0] = 00
#
# -> Printing contents and state of each valid cache line.
# Set: 0, Way: 0, MESI: E, Tag: 000
# Set: 0, Way: 1, MESI: S, Tag: 002
# Set: 0, Way: 2, MESI: M, Tag: 004
# Set: 0, Way: 3, MESI: E, Tag: 006
# Set: 0, Way: 4, MESI: S, Tag: 008
# Set: 0, Way: 5, MESI: M, Tag: 00a
```

```
# Parsed: n = 0,
# Address: Tag[31:20] = 000, Index[19:6] = 0000, Offset[5:0] = 00
#
# -> Read request from L1 data cache, Address: 00000000
# Cache hit for address 0 | In way 0
# Updating for Way:           0
# PLRU BitUpdated PLRU[          0] = 0
# PLRU BitUpdated PLRU[          1] = 0
# PLRU BitUpdated PLRU[          3] = 0
# PLRU BitUpdated PLRU[          7] = 0
# L2: SENDLINE 00000000
# Parsed: n = 0,
# Address: Tag[31:20] = 002, Index[19:6] = 0000, Offset[5:0] = 00
#
# -> Read request from L1 data cache, Address: 00200000
# Cache hit for address 200000 | In way 1
# Updating for Way:           1
# PLRU BitUpdated PLRU[          0] = 0
# PLRU BitUpdated PLRU[          1] = 0
# PLRU BitUpdated PLRU[          3] = 0
# PLRU BitUpdated PLRU[          7] = 1
# L2: SENDLINE 00200000
# Parsed: n = 0,
# Address: Tag[31:20] = 004, Index[19:6] = 0000, Offset[5:0] = 00
#
# -> Read request from L1 data cache, Address: 00400000
# Cache hit for address 400000 | In way 2
# Updating for Way:           2
# PLRU BitUpdated PLRU[          0] = 0
# PLRU BitUpdated PLRU[          1] = 0
# PLRU BitUpdated PLRU[          3] = 1
# PLRU BitUpdated PLRU[          8] = 0
# L2: SENDLINE 00400000
# Parsed: n = 1,
# Address: Tag[31:20] = 006, Index[19:6] = 0000, Offset[5:0] = 00
#
# -> Write request from L1 data cache, Address: 00600000
# Cache hit for address 600000 | In way 3
# Updating for Way:           3
# PLRU BitUpdated PLRU[          0] = 0
# PLRU BitUpdated PLRU[          1] = 0
# PLRU BitUpdated PLRU[          3] = 1
# PLRU BitUpdated PLRU[          8] = 1
# BusOp: INVALIDATE, Address: 00600000, Snoop Result: HIT
# Updated MESI state is M
# Parsed: n = 1,
# Address: Tag[31:20] = 008, Index[19:6] = 0000, Offset[5:0] = 00
```

```
#
# -> Write request from L1 data cache, Address: 00800000
# Cache hit for address 800000 | In way 4
# Updating for Way:           4
# PLRU BitUpdated PLRU[          0] = 0
# PLRU BitUpdated PLRU[          1] = 1
# PLRU BitUpdated PLRU[          4] = 0
# PLRU BitUpdated PLRU[          9] = 0
# BusOp: INVALIDATE, Address: 00800000, Snoop Result: HIT
# Updated MESI state is M
# Parsed: n = 1,
# Address: Tag[31:20] = 00a, Index[19:6] = 0000, Offset[5:0] = 00
#
# -> Write request from L1 data cache, Address: 00a00000
# Cache hit for address a00000 | In way 5
# Updating for Way:           5
# PLRU BitUpdated PLRU[          0] = 0
# PLRU BitUpdated PLRU[          1] = 1
# PLRU BitUpdated PLRU[          4] = 0
# PLRU BitUpdated PLRU[          9] = 1
# Updated MESI state is M
# Parsed: n = 9,
# Address: Tag[31:20] = 000, Index[19:6] = 0000, Offset[5:0] = 00
#
# -> Printing contents and state of each valid cache line.
# Set: 0, Way: 0, MESI: E, Tag: 000
# Set: 0, Way: 1, MESI: S, Tag: 002
# Set: 0, Way: 2, MESI: M, Tag: 004
# Set: 0, Way: 3, MESI: M, Tag: 006
# Set: 0, Way: 4, MESI: M, Tag: 008
# Set: 0, Way: 5, MESI: M, Tag: 00a
#
# No. of cache hits = 6
# No. of cache misses = 6
# No. of cache writes = 5
# No. of cache reads = 7
#
# Finished reading from traces/t17.din.
# ======================================
# ANKARA MESSI ANKARA MESSI MESSI MESSI ANKARA MESSI GOAAAAAAAAAAAAAAL
# Cache Hit Ratio = 0.500000
```
