Electrical and Computer Engineering Department

Portland State University Winter – 2023

(ECE-485/585)
Microprocessor System Design

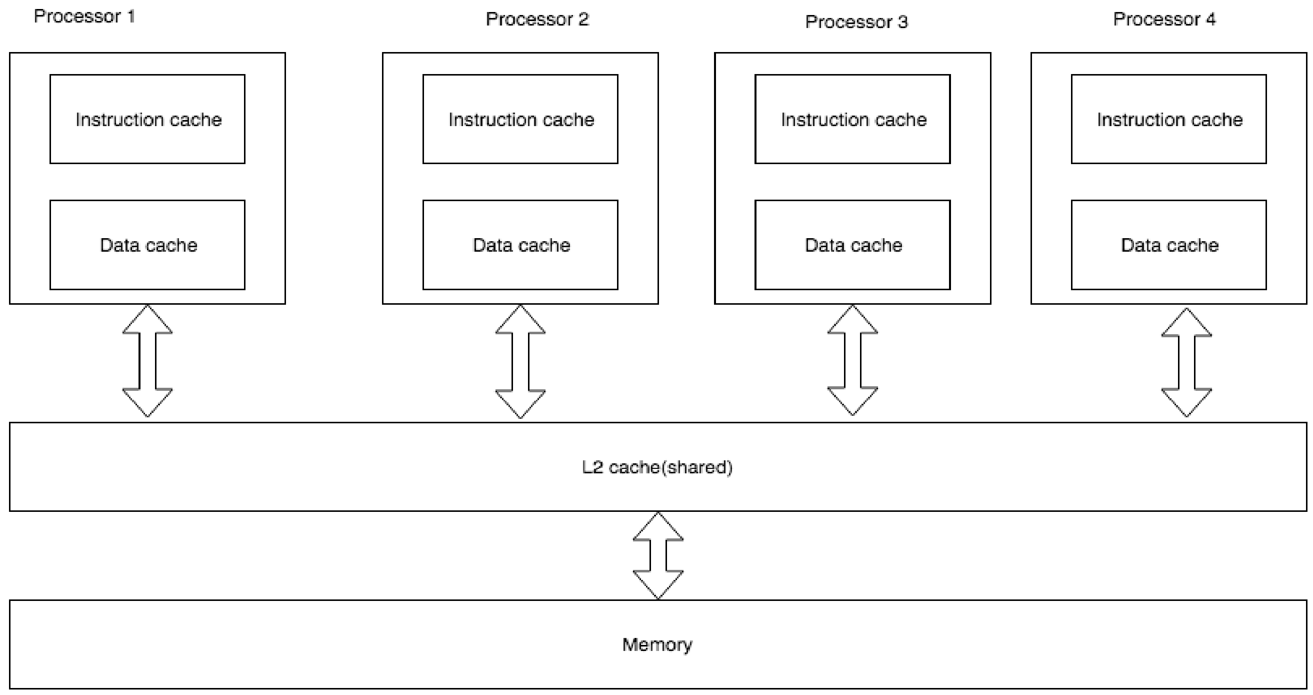
L1-Split Cache Using MESI Coherence Protocol

Project Presented by:

1. Bhuvan Yadav Nagulla
2. Jayanth Kumar Reddy Somu
3. Venkatesh Pamidi
4. Sai Bhargav Reddy Gujjula

1.Introduction:

When the processor requests information that is kept in the cache, a cache is a component in the computing world that provides it to the processor relatively rapidly. These caches typically share high-speed memory components that are located extremely close to the processor to swiftly fulfill all the processor's memory requests. The hardware controls the information in the cache, and the programmer is unable to alter it. In this project, a shared L2 cache memory and a split L1 cache are simulated. The objective is to design and simulate an L1 split cache for a new 32-bit processor with shared memory that uses the MESI protocol to assure cache coherence and can work with up to three other processors.



The L1 instruction cache is four-way set associative and consists of 16K sets and 64-byte lines, and the L1 data cache is eight-way set associative and consists of 16K sets of 64-byte lines. The L1 data cache is write back using write allocate andis write back except for the first write to a line which is write through. Both caches employ LRU replacement policy and are backed by a shared L2 cache. For the cache hierarchy to maintain inclusivity messages using the MESI protocol are sent between L1 and L2 cache.

We have chosen to implement the design in System Verilog. The design is not synthesizable, but it maintains and reportskey cache usage for each cache and display them upon completion of execution of each trace execution. The following statistics reported:

- Number of cache reads
- Number of cache writes
- Number of cache hits
- Number of cache misses
- Cache hit ratio

Note: The above statistics will be reported individually for instruction and data cache separately.

2. Cache Addressing:

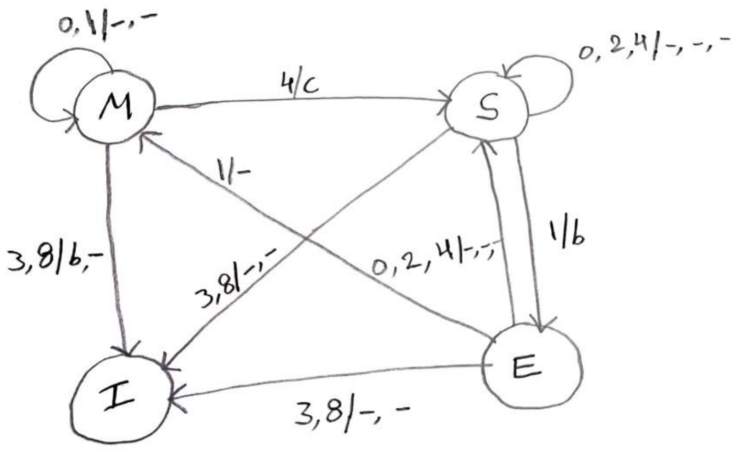
The total number of Address bits = 32 bits.
As each cache line consists of 64 Bytes, the number of bits required for Byte select = 6 bits. As L1 cache consist of 16K sets, The Number of bits required to represent each set = 14 bits.No. of Address Tag bits = 32- (14 + 6) = 12 bits.

Address Tag bits [31:20]	Index bits [19:6]	Byte Offset [5:0]

3. Assumptions:

- Memory references do not extend across cache line boundaries.
- The L2 cache is not designed (it is assumed to be exist and messages are sent).
- All the messages displayed are from L1 cache.
- Only address of the cache is simulated and data is not considered.
- The L2 cache is assumed to be correct.
- The L1 cache employs True LRU replacement policy.
- Cache hierarchy employs inclusivity.
- There will be no MESI state transition from Modified to Shared (any reads in Modified state are assumedto be from acting processor)
- Any reads in the Exclusive state are assumed to be from another processor and the state is changed toShared (is the only case where read/write is assumed to be from another processor)
- Cache hit/miss only happens for read and write operations.
- Write through happens only on cache miss for operation, after write through operation MESI state willbe Exclusive.

4. MESI State Transition Diagram:



Inputs	
0	Read Data Request from L1 cache
1	Write Data Request from L1 cache
2	Instruction fetch
3	Send Invalidate Command from L2
4	Data Request From L2
5	Clear Cache
6	Read Data Request from L1 cache
Outputs	
a	Read from L2 <Address)
b	Write to L2 <Address>
c	Return data to l2<Address>
d	Read for Ownership from L2<Address>

Note: State diagram represents only on cache hits.

Read from L2 <Address> is displayed on read operation and is cache miss

Read for Ownership from L2<Address> is displayed on write operation and is cache miss

5. Pseudo Code:

1. Define parameters for the number of sets, number of Address bits, number of ways in Data cache, number of ways in Instructioncache and number of Cache lines.
2. Define local parameters for the Number of Index bits, Byte select bits, Tag bits, and the Number of bits required for LRU (3 for Datacache and 2 for Instruction cache)
3. Create a typedef Enum with MESI states that requires 2 bits.
4. Define a packed structure typedef for each cache line in the Data and Instruction caches separately (MESI bits, LRU bits, Index bits,Tag bits).
5. Declare Data cache with 16K sets and each set having 8 cache lines, Instruction cache with 16k sets and each set having 4 cachelines.
6. write individual task for different operations in cache

```
//----------------------------------------------------------------------------------------------------------
//Read to processor Data cache and instruction fetch to Instruction cache (0,2)
//----------------------------------------------------------------------------------------------------------

if cache Read (data cache/ Instruction cache)
        increment Read counter
        check for Address with Valid
        statesif (Hit)
                increment Hit count
                Update LRU
                if (MESI state is EXCLUSIVE)
                        update MESI state to SHARED                    //Read from another processor
                else
                        maintain previous MESI state
        else

                increment Miss count
                Check for any cache line with
                INVALIDif (Invalid found)
                        allocate invalid line to new read
                        Read form L2 cache to L1
                        update LRU
                        update MESI state to EXCLUSIVE
                else
                        check for least recently used cache in the set to evict
                        if (evicted cache line MESI state is MODIFIED)
                                write to L2 cache
                                Read form L2 cache to
                                L1Update LRU
                                Update MESI state to EXCLUSIVE
                        else
                                Read from L2 cache to
                                L1Update LRU
                                Update MESI state to EXCLUSIVE
```

```
//---------------------------------------------------------------------------
//Write to processors cache (only for Data cache) (1)
//---------------------------------------------------------------------------

if (cache write)
        increment Write count
        check for Address with valid
        statesif (Hit)
                increment Hit count
                Update LRU
                if MESI state is shared
                        Write to L2 cache
                        update MESI state to EXCLUSIVE
                else
                        update MESI state to MODIFIED                  //write back on cache hit
        else

                increment Miss count
                Check for any cache line with INVALID
                stateif (Invalid found)
                        allocate invalid line to new
                        writeRead for ownership from
                        L2 write L2 address
                        update LRU bits
                        update MESI state to EXCLUSIVE state           //first write is write through
                else
                        check for least recently used cache in the set to evict
                        if (evicted cache line MESI state is MODIFIED)
                                write to L2 cache
                                Read for ownership from L2
                                write to L2 cache for new write
                                Update LRU bits
                                Update MESI state to EXCLUSIVE         //first write is write through
                        else
                                Update LRU bits
                                Update MESI state to EXCLUSIVE state   //first write is write through


//-------------------------------------------------------------------
//Send Invalidate command from L2 address (3)
//-------------------------------------------------------------------
        check for Address with valid states

        if
        (hit)   if (MESI state is MODIFIED)
                        write to L2 cache
                        update MESI state to INVALID
                else
                        update MESI state to INVALID
        else
```

```
            do nothing

//---------------------------------------------------------------
//Data request from L2 cache (snooping) (4)
//---------------------------------------------------------------
        Check for Address with valid states

        if
        (hit)   if (MESI state is MODIFIED)
                    Return data to L2 address
                    update MESI state to
                    INVALID
                else if (MESI state is EXCLUSIVE)
                    Update MESI state to
                    SHARED
                else
                    Maintain previous MESI state
        else

                do nothing
```

6. Traces and Expected Results:

Trace file data which we used to perform all the operations for a Data Cache, Instruction cache:

```
0 984DE132
0 116DE12F
1 100DE130
1 999DE12E
1 645DE10A
0 846DE107
1 211DE128
0 777DE133
0 999DE132
1 116DE123
1 666DE135
1 333DE12C
0 846DE10C
0 777DE136
1 ABCDE128
0 116DE101
1 100DE101
1 AAADE101
1 EDCDE101
2 116DE101
2 100DE101
2 AAADE101
2 EDCDE101
4 AAADE101
0 984DE232
0 116DE22F
1 100DE230
1 999DE22E
1 645DE20A
0 846DE207
1 211DE228
0 777DE233
0 999DE232
1 116DE223
1 666DE235
1 333DE22C
0 846DE20C
0 777DE236
1 ABCDE228
0 116DE201
1 100DE201
1 AAADE201
1 EDCDE201
2 116DE101
2 100DE201
2 AAADE201
2 EDCDE201
4 AAADE201
9 0000
```

Expected Data Cache contents and MESI states:

For index = 14’h00003784:

Way	Tag	MESI State	LRU
0	12'hedc	Modified	3'b111
1	12'h116	Modified	3'b100
2	12'h333	Modified	3'b000
3	-	Invalid	-
4	12'habc	Modified	3'b011
5	12'h846	Shared	3'b001
6	12'h100	Modified	3'b101
7	12'h777	Shared	3'b010

For index = 14’h00003788:

Way	Tag	MESI State	LRU
0	12'hedc	Modified	3'b111
1	12'h116	Modified	3'b100
2	12'h333	Modified	3'b000
3	-	Invalid	-
4	12'habc	Modified	3'b011
5	12'h846	Shared	3'b001
6	12'h100	Modified	3'b101
7	12'h777	Shared	3'b010

Data Cache Reads=16
Data Cache Writes=22
Data Cache Hits =10
Data Cache Misses= 28
Data Cache Hit Ratio = 26.315789

Expected Instruction Cache contents and MESI states:

For index = 14’h00003784:

Way	Tag	MESI State	LRU
0	12'h116	Shared	2'b11
1	12'h100	Exclusive	2'b00
2	12'haaa	Exclusive	2'b01
3	12'hedc	Exclusive	2’b10

For index = 14’h00003788:

Way	**Tag**	**MESI State**	**LRU**
0	12'h100	Exclusive	2’b01
1	12'haaa	Exclusive	2'b10
2	12'hedc	Exclusive	2'b11

Instruction Cache Reads = 8
Instruction Cache Misses=7
Instruction Cache Hits=1
Instruction Cache Hit Ratio = 12.500000

7. Attachments:

L1Cache.sv

output.txt

trace.txt

Project_Team25_Report.pdf

Project Roles & Responsibility Team-25.pdf