ECSE 6700 Project 1 Multi-Level Cache

Rui Xie (xier2@rpi.edu)

02/18/2024

1 Introduction

1.1 Objective

The primary goal of this project is to design and implement a multi-level cache system, which includes a two-level cache hierarchy with L1 and L2 caches and an arbiter for managing memory access requests. This system is equipped with the MESI protocol for cache coherence, ensuring consistency in a multi-core processing environment.

1.2 Background

Caches are a critical component in modern computing architectures, providing faster access to data by bridging the speed gap between the processor and the main memory. A multi-level cache further enhances performance by reducing latency and improving hit rate. The MESI (Modified, Exclusive, Shared, Invalid) protocol is employed for maintaining coherence across multiple caches, which is essential in multi-core systems.

2 Cache Framework

The design of the multi-level cache system in this project focuses on a two-tier architecture, comprising L1 cache and L2 cache. Fig. 1 shows the illustration of the design framework.

L1 Cache: The L1 cache in this system is configured with a size of 256 bytes and a line size of 4 bytes. The L1 cache handles processor read and write requests, checking its entries for data corresponding to the processor's requested memory addresses. In the event of a cache hit, where the requested data is found within the cache, the data is promptly provided to the processor. In cases of a cache miss, the request is forwarded to the L2 cache for further processing.

L2 Cache: The L2 cache is configured with 1024 bytes and the same line size as the L1 cache, and employs a 4-way set associative design. The L2 cache utilizes the Least Recently Used (LRU) policy for eviction decisions. This policy ensures that the data least recently accessed is replaced when new data needs

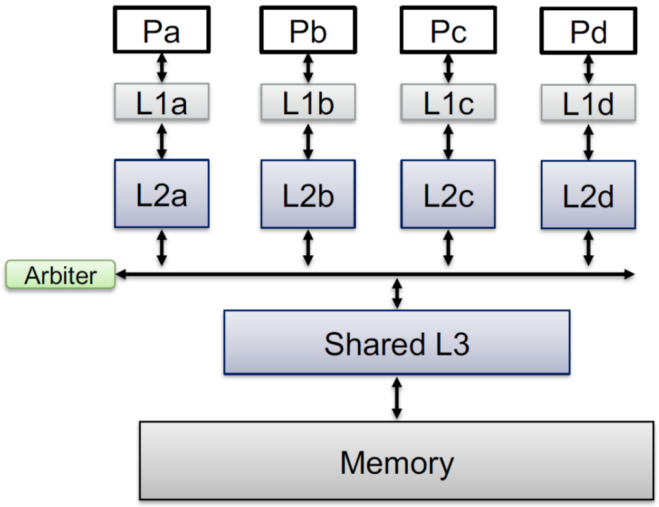

Figure 1: Illustration of multi-level cache design, we initiate 4 processors and each processor has private L1 and L2 cache. All L2 cache modules are connected to the next level shared memory.

to be loaded into a full cache set, maintaining a balance between recently and less frequently used data.

MESI Protocol (Cache Coherence): The implementation of the MESI protocol is instrumental in ensuring consistency across both L1 and L2 caches. The protocol manages four states for each cache line: Modified, Exclusive, Shared, and Invalid. This state management allows the system to effectively handle various scenarios, such as simultaneous data requests from multiple processors and updates to shared data blocks. The protocol's snooping mechanism enables each cache to monitor and respond to access requests, thereby maintaining a consistent state of data across the cache hierarchy.

Arbiter (Managing Memory Access): The arbiter module is a crucial component that manages memory access requests from multiple L2 caches. In scenarios where multiple caches simultaneously request access to the shared main memory, the arbiter ensures fair and efficient resolution of these requests. It arbitrates among competing requests, granting access based on a predefined strategy, which could be as simple as a round-robin or a priority-based scheme. This mechanism prevents conflicts and ensures efficient utilization of the memory bandwidth.

3 Experiment

3.1 Simple Read Operation (L1 Cache Instance 0)

1. Setup: Initialized L1 cache instance 0 for a read operation.
2. Operation: A read request was triggered with a specific memory address (32'h0000_0004).
3. Observations: The test monitored the cache's response to the read request. It checked if the data was retrieved successfully from the cache or fetched from the L2 cache or memory in case of a miss. The MESI state of the cache line post-operation was also observed.
4. Expected Outcome: Accurate data retrieval with appropriate MESI state transition, indicating efficient cache hit or miss handling.

3.2 Simple Write Operation (L1 Cache Instance 1)

1. Setup: Configured L1 cache instance 1 for a write operation.
2. Operation: Executed a write request to a specific memory address (32'h0000_0008) with the data 32'h1234_5678.
3. Observations: The system's handling of the write operation was examined, including how the data was stored in the cache and whether it triggered any MESI state changes. The coherence of this write operation with other cache instances was also considered.
4. Expected Outcome: Successful data storage in the cache with correct MESI state update, demonstrating effective write operation handling.

3.3 Cache Miss and Fetch from Memory (L1 Cache Instance 2)

1. Setup: Prepared L1 cache instance 2 to simulate a cache miss scenario.
2. Operation: Initiated a read request to an address (32'h0000_1000) likely to result in a cache miss.
3. Observations: The behavior of the cache system during a miss was analyzed, particularly its interaction with the L2 cache and memory to fetch the requested data. The efficiency of the cache in handling misses and the MESI protocol's role in this scenario were key focus areas.
4. Expected Outcome: Efficient handling of the cache miss, including correct data fetching from memory and appropriate MESI state management.

3.4 Cache Coherence Check among all L1 Cache Instances

1. Setup: Engaged all L1 cache instances to test cache coherence.
2. Operation: Performed write operations to a common address (32'h0000_000C) from different instances, each writing unique data. This was followed by read operations to verify if the data in all caches remained coherent.
3. Observations: The test closely monitored the MESI state transitions across all cache instances and verified if the data remained consistent and coherent post-multiple write operations.
4. Expected Outcome: Consistent and coherent data across all L1 cache instances, showcasing the effective implementation of the MESI protocol in maintaining cache coherence.

As depicted in Fig. 2, the waveform from our four test cases is clearly illustrated, aligning well with our anticipated results.

4 Conclusion

This project successfully demonstrates the design and functionality of a multi-level cache system using SystemVerilog. The implementation effectively uses the MESI protocol to ensure data coherence in a multi-core environment, showcasing the importance of sophisticated cache systems in modern computing architectures. The experiments conducted validate the system's performance, coherence, and efficiency in handling various cache operations.

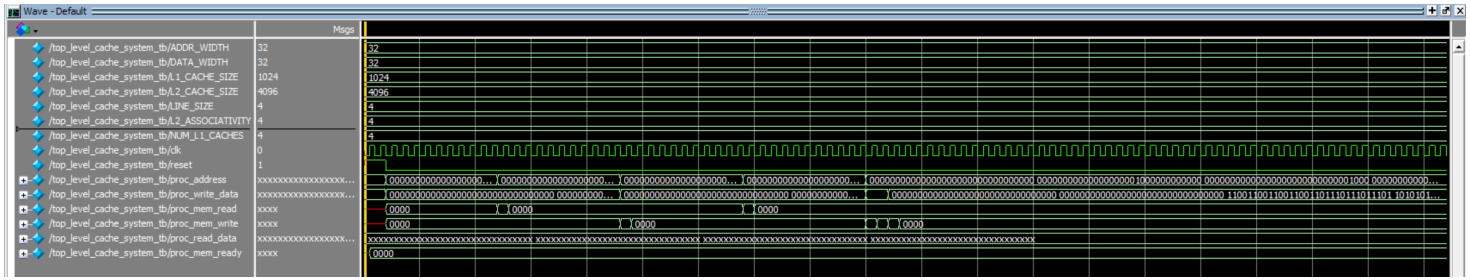

Figure 2: Waveform of the 4 cases in testbench.