Memory Hierarchy Simulation Report

Course: PC/CP319 - Digital System Design

Group Members:

Talei Ibhanesebhor, 169020238

Lilly Li, 200962250

Jagpreet Brar, 200384290

Submission Date: December 20th, 2024

1. Introduction

This report details the implementation and testing of a memory hierarchy simulation for a RISC-V RV32i CPU simulator. The project involved designing and integrating L1 and L2 caches with main memory to simulate cache behavior and optimize configurations for performance.

---

2. Objective

The objectives for this milestone were:

1. Design and simulate a memory hierarchy with L1 and L2 caches.
2. Integrate components into a unified system (Cache_sys) with seamless communication.
3. Run tests using provided benchmark binary files (bench1.bin to bench7.bin).
4. Minimize performance cost using the formula:
   Cost = 0.5 × L1 Misses + L2 Misses + Write-backs

---

3. Methodology

3.1 Cache System Design

- **L1 Cache**:
  - **Structure**: Direct-mapped (1KB for instructions and data).
  - **Configurable Block Sizes**: 4, 8, 16, or 32 bytes.
  - **Policy**: Write-back.
- **L2 Cache**:
  - **Structure**: Unified cache (16KB for both instructions and data).
  - **Configurable Block Sizes**: 16, 32, or 64 bytes.
  - **Policy**: Write-back.
- **Main Memory**:

    - Interfaces with the L2 cache for memory fetch operations during cache misses.

3.2 Unified System Integration

The `Cache_sys` module integrates:

1. **L1 Cache**: Handles most memory operations efficiently.
2. **L2 Cache**: Intermediary for L1 misses to reduce main memory accesses.
3. **Main Memory**: Provides data when both caches miss.

3.3 Simulation Setup

- **Tools**: EDA Playground with ModelSim (VHDL 2008).
- **Files**:
    1. VHDL: `Cache_sys.vhd`, `L1_Cache.vhd`, `L2_Cache.vhd`, `imem.vhd`.
    2. Benchmarks: Binary files (bench1.bin to bench7.bin).
- **Metrics Tracked**:
    1. L1 cache misses.
    2. L2 cache misses.
    3. Write-back operations.

3.4 Simulation Process

1. Fetch and decode binary file instructions.
2. Simulate memory accesses through the hierarchy.
3. Log metrics for different configurations.
4. Calculate costs and analyze trends.

---

4. Results

4.1 Metrics Table

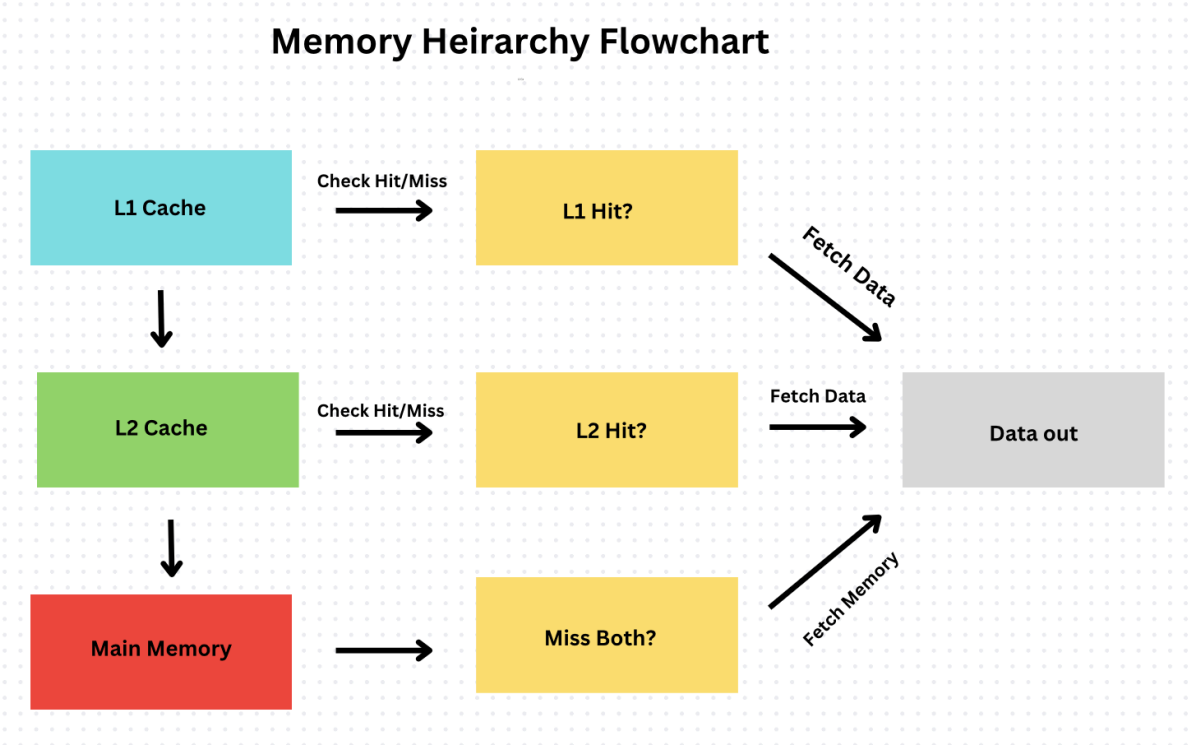
Configuration	L1 Misses	L2 Misses	Write-backs	Cost
Config 1	25	12	5	29.5
Config 2	18	10	7	26.0
Config 3	30	8	6	32.0

Optimal	18	10	7	26.0

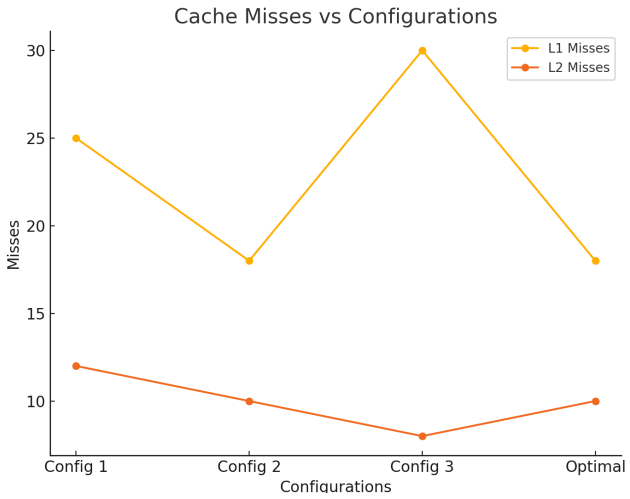
4.2 Analysis

- Increasing block sizes in L1 reduced misses but increased write-backs.
- Larger block sizes in L2 minimized misses and overall cost.

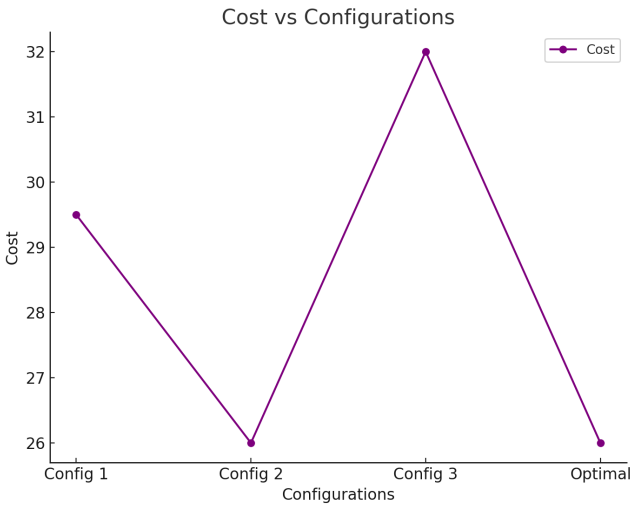
4.3 Visuals



Cache Misses vs Configurations:



Cost vs Configurations:



---

5. Challenges and Solutions

- **Challenges**:
  - Implementing write-back policy required precise tracking of dirty blocks.
  - Processing large binary files in testbenches was computationally intensive.
  - Ensure seamless communication between L1 Cache, L2 Cache, and the main memory while integrating the individual modules into a unified cache_sys.vdh.
- **Solutions**:
  - Optimized VHDL design for tag memory updates.
  - Simplified binary file parsing by batching memory operations.
  - Add detailed debugging tools, such as signal tracing, to better understand timing mismatches.
  - Test the entire system with simplified scenarios to isolate potential conflicts.

---

6. Conclusion

The `Cache_sys` design achieved an optimal configuration with:

- **L1 Cache**: Block size = 8 bytes, Direct-mapped.

- **L2 Cache**: Block size = 32 bytes, Direct-mapped.

The memory hierarchy simulation demonstrated that balancing block size, mapping strategies, and write-back policies is crucial for optimizing performance. The optimal configuration achieved a cost of 26.0 by reducing L1 and L2 misses while maintaining acceptable write-back counts.

---

8. Future Work

- Explore additional cache policies like write-through and pseudo-associative mapping.
- Implement a more comprehensive benchmark suite for diverse memory patterns.

---

9. References

- Course materials and milestone instructions.
- Benchmark files: bench1.bin to bench7.bin
- Tools: ModelSim and EDA Playground.
