DSD HW3

林桓鈺 B10502013

Cycle Time

- **In cache_syn.sdc:** 4.99.
- **In tb_cache.v:** 7.5.

Direct-mapped Cache

General Specification of Cache Unit

- **Total Cache Size**: 256 words
- **Cache Line Size**: Each cache line holds 32 words (128 bytes)
- **Number of Cache Lines**: 8 lines
- **Word Size**: Each word is 32 bits (4 bytes)
- **Associativity:** 1 (direct-mapped)

Placement Policy:

- **Direct Mapped Cache**: Each memory block maps to exactly one cache line. This is determined by the `INDEX` bits of the address.
- **Tag, Index, and Offset**:
  - **Tag**: 25 bits
  - **Index**: 3 bits
  - **Offset**: 2 bits

Read/Write Policy:

- **Write-Back**: Dirty data is written back to memory when replaced.

Cache States:

- **IDLE**: Cache is idle.
- **Memory Read**: Cache is fetching data from memory.
- **Memory Write-Back**: Cache is writing back dirty data to memory.
- **Write**: Cache is handling a write operation.

Cache Operations:

- **Read Hit**: Data is found in the cache.
- **Read Miss**: Data is not found in the cache and needs to be fetched from memory.
- **Write Hit**: Data is found in the cache and is being modified.
- **Write Miss**: Data is not found in the cache, so it is fetched from memory and then modified.

Two-way Associative Cache

Cache Size and Organization:

- **Total Cache Size**: 256 words
- **Cache Line Size**: Each cache line holds 32 words (128 bytes)
- **Number of Sets**: 4 sets (2 lines per set)
- **Associativity**: 2-way set-associative

Placement Policy:

- **LRU (Least Recently Used)**: When data1 in the set is used, we set `out_data = 0` which means to replace the data0 because data1 is least recently used.
- **Tag, Index, and Offset**:
  - **Tag**: 26 bits
  - **Index**: 2 bits
  - **Offset**: 2 bits

Read/Write Policy:

- **Write-Back**: Dirty data is written back to memory when replaced.

Cache States:

- Same as direct-mapped cache

Cache Operation:

- Same as direct-mapped cache.

Performance Evaluation

Direct-mapped Cache:

- **First read operation:**
  - **Miss rate:** 25%.
  - **Execution cycle:** 2560.
  - **Stall cycle:** 1536.
- **Write operation:**
  - **Miss rate:** 25%.
  - **Execution cycle:** 3800.
  - **Stall cycle:** 2766.
- **Second read operation:**
  - **Miss rate:** 100%.
  - **Execution cycle:** 7208.
  - **Stall cycle:** 6184.

Two-way Associative Cache:

- **First read operation:**
  - **Miss rate:** 25%.
  - **Execution cycle:** 2560.

    - **Stall cycle:** 1536.
- **Write operation:**
    - **Miss rate:** 25%.
    - **Execution cycle:** 3820.
    - **Stall cycle:** 2796.
- **Second read operation:**
    - **Miss rate:** 62.5%.
    - **Execution cycle:** 4884.
    - **Stall cycle:** 3860.

Discussion

- Two types of cache perform same in first read operation. But in second read operation, the two-way associative cache perform better. This indicates that the two-way associative cache provides a better hit rate compared to the direct-mapped cache for subsequent reads due to its increased flexibility in storing data.

1. **Miss Rate:**
    - In the direct-mapped cache, the miss rate for the second read operation increases to 100%, while in the two-way associative cache, it increases to 62.5%. This indicates that the two-way associative cache provides a better hit rate compared to the direct-mapped cache for subsequent reads due to its increased flexibility in storing data.
2. **Execution Cycles:**
    - The execution cycles for write operations are slightly higher in the two-way associative cache. This is because, in the two-way associative cache, additional checks need to be made to determine which block to replace in case of a write.
3. **Stall Cycles:**

- In the direct-mapped cache, the stall cycles for the second read operation are significantly higher compared to the two-way associative cache. This is because the direct-mapped cache cannot hold multiple blocks for the same index, causing frequent cache misses and stall cycles.
- The higher stall cycles in the direct-mapped cache also contribute to the higher execution cycles observed for the second read operation.

Conclusion

During this assignment, I spent some time watching videos and reading lectures on Cool because I wasn't familiar with computer architecture. When writing the program, I encountered many difficulties, such as not knowing when certain signals should be high or low, so I spending much of time to figure out all the details. I sought help from classmates, and with their assistance, I managed to complete the assignment. I realize that I should have taken a computer architecture course before enrolling in this one. I overestimated my abilities at the beginning of the semester, which made it difficult for me to prepare for the mid-term exam and complete assignments later on. I hope to get through this semester smoothly.