Project 0 Simple Cache

Rui Xie (xier2@rpi.edu)

01/18/2024

1 Introduction

This homework is to design a simple cache controller.

2 Cache Framework

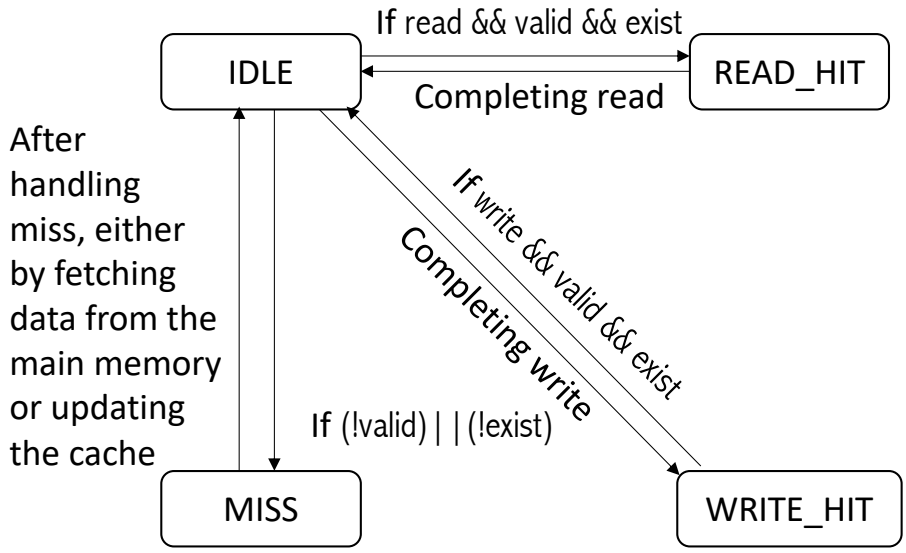

Figure 1: Cache controller FSM diagram. States: IDLE (waiting for requests), READ_HIT (reading data from cache), WRITE_HIT (updating cache data), and MISS (handling cache misses). Transitions are based on CPU read/write requests and cache hit/miss status.

Fig. 1 shows the diagram of the Finite State Machine (FSM) diagram. In IDLE, the controller awaits CPU read or write requests. READ_HIT occurs on a read request hit, where data is read from the cache. WRITE_HIT is entered on

a write request hit, updating data in the cache. In MISS, the controller handles cache misses by fetching data from the main memory or updating the cache. Transitions between these states are based on CPU requests and cache hit/miss determinations.

Table 1: Cache controller design parameters

Parameter	Description	Value
DATA_WIDTH	Width of data in the cache line (bits)	32
ADDR_WIDTH	Width of the memory address (bits)	32
NUM_LINES	Number of lines in the cache	256
TAG_WIDTH	Width of the tag in the cache line	10

Table 1 shows the key parameters of the cache controller, including data width, address width, number of cache lines, and tag width.

The simple_cache_controller module in SystemVerilog features key inputs and outputs essential for its operation. The addr input specifies the memory address for data access, and write_data carries the data to be written into the cache. Important control signals include read and write, which dictate the type of operation (read or write) to be performed by the controller. On the output side, read_data provides the data retrieved from the cache, while hit and miss signals indicate the success or failure (respectively) of a cache access operation. These components collectively ensure efficient and effective cache memory management by the controller.

3 Experiment

To verify the behavior of the simple cache controller, we simulate read and write operations with different addresses and observe the cache hit/miss status and data integrity. The following test cases are designed to validate the functionality of the simple_cache_controller module implemented in SystemVerilog.

1. **Test Case 1 (Read from an Unwritten Address - Expect MISS):** Read from an unwritten address `32'hC`. Expected outcome: Cache miss signal (`miss`) is asserted.

2. **Test Case 2 (Write then Read from a Different Address - Expect MISS):** Write data `32'hCAFEBABE` to address `32'hE` and then read from a different address `32'hB`. Expected outcome: Cache miss signal (`miss`) is asserted.

3. **Test Case 3 (Write then Read from the Same Address - Expect HIT):** Write data `32'hBABECAFE` to address `32'hD` and then read from the same address. Expected outcome: Cache hit signal (`hit`) is asserted.

4. **Test Case 4 (Write to an Existing Address - Expect HIT on Read):** Write new data `32'h12345678` to a previously written address

`32'hD`, then read from the same address. Expected outcome: Cache hit signal (`hit`) is asserted.

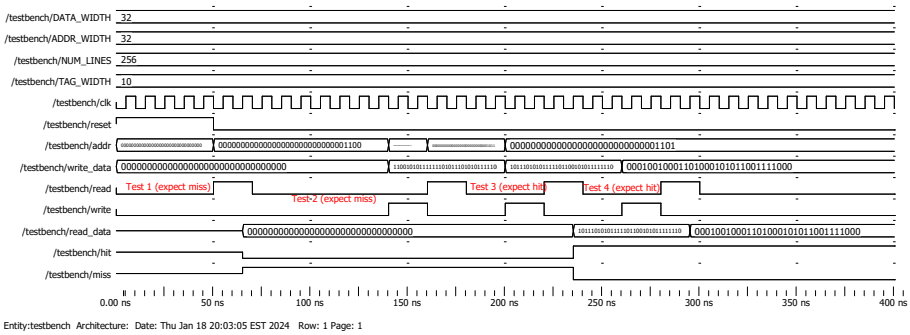

Figure 2: Waveform of the experiment.

Fig. 2 clearly shows the results of hit and read signals, each case is highlighted in red.