**Cache Controller Documentation**

**Overview**

The cache_controller module implements a basic cache controller for a direct-mapped cache system. The controller manages cache operations, including cache read and write requests, cache allocation, and write-back processes. This module interfaces with a simplified internal cache memory and handles requests from a CPU.

**Parameters**

- **DATA_WIDTH**: The width of the data bus. (default: 32 bits)
- **CACHE_LINES**: Number of cache lines in the cache memory. (default: 4096 lines)

**Ports**

**Inputs**

- **clk**: Clock signal (rising edge triggered).
- **reset**: Asynchronous reset signal. Active high.
- **flush**: Signal to flush the cache.
- **address**: 32-bit address for cache operations.
- **data_in**: Data input for write operations.
- **cpu_request**: Structure to specify CPU request type, including read and write requests.

**Outputs**

- **hit**: Signal indicating a cache hit.
- **miss**: Signal indicating a cache miss.
- **data_out**: 32-bit data output for read operations.
- **read_done**: Signal indicating that a read operation is complete.
- **write_done**: Signal indicating that a write operation is complete.
- **busy**: Signal indicating that the cache controller is busy processing a request.

**Internal Signals**

- **address_fields**: Structure to hold the tag, index, and offset extracted from the address.
- **flush_done**: Internal signal used to indicate when a flush operation is complete.

**States**

The cache controller operates in the following states:

- **IDLE**: The cache controller is idle and waiting for a request.
- **PROCESS_REQUEST**: The controller processes the incoming cache request.
- **CACHE_ALLOCATE**: The controller allocates a cache line and updates the cache.
- **WRITE_BACK**: The controller writes dirty cache data back to memory.
- **FLUSH**: The controller performs cache flush operations.

**Cache Memory**

- **cache_mem**: Array of cache_line_type representing the cache memory. Each cache line includes:
    - **valid_bit**: Indicates if the cache line contains valid data.
    - **dirty_bit**: Indicates if the cache line has been modified.
    - **tag**: Tag stored in the cache line.
    - **data**: Data stored in the cache line, 128 bits wide.

**Functional Description**

1. **Address Extraction**:
    - Extracts the tag, index, and offset from the 32-bit address.
2. **State Machine**:
    - **IDLE**: Waits for a read or write request. Transitions to PROCESS_REQUEST when a request is received.
    - **PROCESS_REQUEST**: Checks if the cache line is valid and the tag matches:
        - If a cache hit and read request: Outputs data and signals read completion.
        - If a cache miss: Determines whether to allocate a new cache line or perform a write-back.
    - **CACHE_ALLOCATE**: Updates the cache with new data. Handles both read and write operations by simulating main memory interaction.
    - **WRITE_BACK**: Writes modified data from the cache line back to memory (simulated in this implementation).
    - **FLUSH**: Implements cache flush operations if needed (currently not detailed).

**State Transition and Memory Operations**

- **State Transitions**:
    - Managed based on current state and request type.

    - Transitions between states are handled on the rising edge of the clock.

- **Memory Operations**:
    - Simulated by updating internal cache memory.
    - Read and write operations are managed by updating the cache line's data, tag, valid_bit, and dirty_bit.

**Testbench**

The provided testbench initializes the cache controller, generates clock signals, and applies various read and write requests to verify functionality. It monitors the cache controller's responses and ensures correct operation.

**Example Test Scenarios**

1. **Write Operation**:
    - Provide an address and data.
    - Set the write request.
    - Check if the cache memory is updated and write_done is asserted.
2. **Read Operation**:
    - Provide an address.
    - Set the read request.
    - Check if the correct data is output and read_done is asserted.
3. **Cache Miss Handling**:
    - Verify that a cache miss leads to cache allocation or write-back, as appropriate.

**Summary**

The cache_controller module is designed to handle cache operations, manage cache states, and interact with an internal cache memory. It simulates interactions with main memory for testing purposes. The state machine and memory operations ensure that the cache controller processes requests efficiently and correctly.