**Cache Controller Using HDL**

In modern computing systems, performance is a critical factor that determines the efficiency and speed of data processing. One of the key components that significantly impacts system performance is the memory hierarchy, where the cache memory plays a vital role. A cache is a smaller, faster memory component located between the CPU and the main memory, designed to store frequently accessed data and instructions to reduce the average time to access data from the main memory.

To manage the cache efficiently, a cache controller is used. A cache controller is responsible for handling the operations of reading and writing data between the CPU and the cache, as well as maintaining coherence and consistency with the main memory. It ensures that the most frequently used data is readily available to the CPU, thereby minimizing latency and improving overall system performance.

The design and implementation of a cache controller can be complex, requiring a precise understanding of memory access patterns, data replacement policies, and synchronization mechanisms. To achieve this, Hardware Description Languages (HDLs) such as VHDL (VHSIC Hardware Description Language) and Verilog are widely used. HDLs allow designers to describe the behavior and structure of electronic systems at a high level, making it possible to model, simulate, and synthesize hardware components.

For implementation we developed the following modules:

* Cache Line

**Overview**

The Verilog code includes multiple modules designed to perform bitwise operations, cache line management, and provide a testbench for verification.

**1. and\_wordgate Module**

Purpose: Computes a bitwise AND across all bits of an input vector.

Parameters: w (width of the input vector, default 8).

Inputs: in (input vector of width w).

Outputs: AND\_ (result of AND operation across all bits).

**2. bit\_comparator Module**

Purpose: Compares two single bits and checks if they are equal.

Inputs: b0, b1 (bits to compare).

Outputs: eq (1 if bits are equal, 0 otherwise).

**3. bitwise\_and Module**

Purpose: Performs a bitwise AND operation on two single bits.

Inputs: in\_0, in\_1 (bits to AND).

Outputs: and\_ (result of AND operation).

**4. bitwise\_comparator Module**

Purpose: Compares two vectors bit by bit and checks if all bits are equal.

Parameters: w (width of input vectors, default 8).

Inputs: in\_0, in\_1 (input vectors).

Outputs: eq (1 if all bits are equal, 0 otherwise).

Uses: Multiple bit\_comparator instances and and\_wordgate for final equality check.

**5. cache\_line Module**

Purpose: Manages a single cache line including read, write, age, and hit/miss detection.

Parameters: ADDRESS\_WORD\_SIZE (32), TAG\_SIZE (19), BLOCK\_SIZE (8), WORD\_SIZE (8).

Inputs:

clk, rst\_b (clock and reset signals).

ready, address\_word, try\_read, try\_write, write\_data, reset\_age, increment\_age.

Outputs: data, age, hit\_miss, is\_empty.

**Functionality:**

Uses bitwise\_comparator for tag matching.

Handles read/write operations based on hit/miss status.

Manages cache line age and validity.

**6. cache\_line\_tb Module (Testbench)**

Purpose: Tests the cache\_line module.

Inputs/Outputs: Mimics the cache\_line I/O for testing.

**Functionality**:

Initializes and toggles the clock.

Asserts and deasserts the reset.

Generates random test values for address, operations, and data.

Monitors the outputs to verify correct functionality.

This set of modules, together with the testbench, provides a comprehensive structure for bitwise operations and cache management, ensuring thorough verification of the cache line's functionality.

* **Four Way Set Module:**

**Purpose**:

The four\_way\_set module implements a four-way set associative cache set. It manages multiple cache lines within a set, handles read and write operations, updates cache line ages, and determines cache hits or misses.

**Parameters**:

ADDRESS\_WORD\_SIZE: Width of the address word (default is 32 bits).

TAG\_SIZE: Size of the tag in the address (default is 19 bits).

BLOCK\_SIZE: Size of each cache block (default is 8 bytes).

WORD\_SIZE: Size of a word in the cache (default is 8 bits).

Inputs:

clk: Clock signal.

rst\_b: Active low reset signal.

address\_word: The address for the cache operation.

try\_read: Signal to initiate a read operation.

try\_write: Signal to initiate a write operation.

write\_data: Data to be written to the cache.

reset\_age: Signal to reset the age of specific cache lines.

increment\_age: Signal to increment the age of specific cache lines.

**Outputs**:

data: Data read from the cache.

ages: Age information of the cache lines.

hit\_miss: Indicates if the operation was a hit or miss.

hit\_miss\_set: Indicates which set had the hit or miss.

Internal Components:

**Registers:**

ready: A 4-bit register indicating which cache line is ready for the next operation.

Wires:

line\_data: Concatenated data outputs from the four cache lines (32 bits in total, 8 bits per line).

line\_hit\_miss: Indicates if each cache line had a hit (4 bits).

line\_is\_empty: Indicates if each cache line is empty (4 bits).

sel: The selected cache line for the current operation (2 bits).

**Modules:**

encoder4to2: Encodes the 4-bit ready signal into a 2-bit sel signal.

cache\_line: Four instances, each representing a cache line.

mux4to1: Two instances to select the data and hit/miss signals from the four cache lines.

**Functionality**:

**Initialization**:

The ready signal is initialized to 4'd0, indicating that no cache lines are ready initially.

**Cache Line Management:**

Four instances of the cache\_line module handle the actual data storage and retrieval.

Each cache line operates based on the provided signals (try\_read, try\_write, reset\_age, increment\_age).

**Data and Hit/Miss Selection:**

The mux4to1 modules select the appropriate data and hit/miss signals based on the sel signal generated by the encoder4to2 module.

The hit\_miss\_set output is directly assigned from line\_hit\_miss.

State Updates:

**The always block updates the ready signal based on the hit/miss status and age of the cache lines:**

If a cache line has a hit, it is marked as ready.

If a cache line is empty, it is marked as ready.

If no cache line has a hit and none are empty, the cache line with the oldest age is marked as ready.

This prioritizes cache hits first, then empty lines, and finally the least recently used line if all lines are valid.

four\_way\_set\_tb (Testbench)

**Purpose**:

The testbench verifies the functionality of the four\_way\_set module by simulating various scenarios, including cache reads and writes, and updating ages.

**Signals**:

clk: Clock signal (toggled every half period).

rst\_b: Active low reset signal (asserted for a defined pulse duration and then deasserted).

address\_word: Randomly generated address words for the cache operations.

try\_read: Randomly generated signal to initiate read operations.

try\_write: Randomly generated signal to initiate write operations.

write\_data: Randomly generated data to be written to the cache.

reset\_age: Randomly generated signal to reset the age of a cache line.

increment\_age: Signal to increment the age of a cache line (complement of reset\_age).

**Functionality**:

Simulates a variety of operations, including reads, writes, and age updates, by cycling through different random values for the inputs.

Monitors the outputs (data, ages, hit\_miss, hit\_miss\_set) to verify the correct functionality of the four\_way\_set module.

**Summary**

The four\_way\_set module is an essential part of a set-associative cache, managing four cache lines and handling read/write operations, age management, and hit/miss detection. The testbench rigorously tests this functionality by simulating different operational scenarios, ensuring that the module operates correctly under various conditions. This setup ensures efficient cache management and accurate hit/miss tracking, contributing to the overall performance of the cache system.

* **Cache Controller**

**1. tristate\_driver Module**

Purpose: To conditionally pass or block a signal based on an enable control.

Functionality:

Takes an input vector in of width w and an enable signal enable.

If enable is high (1), the output out is set to the value of in.

If enable is low (0), the output out is set to high impedance ('z), effectively disconnecting it from the circuit.

**2. mux128to1 Module**

Purpose: To select one out of 128 input vectors based on a 7-bit selection signal.

Functionality:

Takes a wide input vector in of size w \* 128 (representing 128 input vectors of width w each) and a 7-bit selection signal sel.

Outputs the vector from in corresponding to the index given by sel.

For example, if sel is 3, the module outputs the third w-bit segment from in.

**3. dec7to128 Module**

Purpose: To decode a 7-bit binary input into a one-hot 128-bit output.

Functionality:

Takes a 7-bit input index.

Sets the bit at the position specified by index to 1, while all other bits are set to 0.

For example, if index is 5, the output active will have the sixth bit set to 1 and all other bits set to 0.

**4. cache\_memory Module**

Purpose: To manage a multi-set cache memory system.

Functionality:

Inputs:

clk, rst\_b: Clock and reset signals.

address\_word: Address for the cache operation.

try\_read, try\_write: Control signals for read and write operations.

write\_data: Data to be written to the cache.

reset\_age, increment\_age: Signals to manage the age of cache lines.

Outputs:

data: Data output from the cache.

ages: Age information of the cache lines.

hit\_miss: Indicates whether the cache operation was a hit or miss.

hit\_miss\_set: Indicates which set had the hit or miss.

Internal Mechanism:

Uses the dec7to128 decoder to activate the appropriate set based on the address.

Employs tristate\_driver modules to route the try\_read, try\_write, reset\_age, and increment\_age signals to the selected set.

Each cache set is managed by a four\_way\_set module (assumed to be defined elsewhere), which handles the actual storage and retrieval of data.

Uses mux128to1 modules to select the appropriate data, age, and hit/miss outputs from the sets based on the address.

**5. cache\_memory\_tb Module (Testbench)**

Purpose: To test the cache\_memory module and ensure its correct functionality.

Functionality:

Signals:

Generates clock (clk) and reset (rst\_b) signals.

Randomly generates address words (address\_word), read/write operations (try\_read, try\_write), and write data (write\_data).

Manages age reset and increment signals (reset\_age, increment\_age).

Testing Procedure:

Initializes the clock to toggle at a specified period.

Asserts the reset signal initially and then deasserts it.

Randomly generates test values for the address, read/write operations, and data to simulate various scenarios.

Monitors the outputs (data, ages, hit\_miss, hit\_miss\_set) to verify the correct functionality of the cache\_memory module.

**Summary**

The collection of modules creates a system for managing cache memory operations with multi-set support. The cache\_memory module orchestrates cache reads and writes, handles data aging, and tracks cache hits and misses. The tristate\_driver and mux128to1 modules enable efficient signal routing and data selection, while the dec7to128 decoder facilitates set activation. The testbench verifies the overall functionality by simulating various operational scenarios.

* **Control Unit**

control\_unit Module

Purpose:

The control\_unit module manages the state and control signals for a cache system. It handles read and write operations, updates cache line ages, and determines the next state based on the current operation and hit/miss status.

**Parameters**:

ADDRESS\_WORD\_SIZE: The width of the address word (default is 32 bits).

Inputs:

clk: Clock signal.

rst\_b: Active low reset signal.

opcode: Operation code (determines read/write).

hit\_miss: Indicates whether the previous operation was a cache hit or miss.

hit\_miss\_set: Set index for hit/miss detection (4 bits).

ages: Age information of cache blocks (8 bits).

Outputs:

address\_word: Address word for the operation.

try\_read: Signal to attempt a read operation.

try\_write: Signal to attempt a write operation.

write\_data: Data to be written (8 bits).

reset\_age: Signal to reset the age of a cache block (4 bits).

increment\_age: Signal to increment the age of a cache block (4 bits).

Internal State:

Local parameters for state definitions: IDLE, TRY\_READ, TRY\_WRITE, STALL\_1, STALL\_2, UPDATE\_AGES, CHECK\_STATUS.

Registers: current\_state, next\_state, replace\_index.

**State Machine:**

IDLE: Initial state. Determines the next state based on the opcode.

If opcode is high, transition to TRY\_WRITE.

If opcode is low, transition to TRY\_READ.

TRY\_READ: Attempts a read operation. Generates a random address and transitions to CHECK\_STATUS.

TRY\_WRITE: Attempts a write operation. Generates a random address and write data, then transitions to CHECK\_STATUS.

STALL\_1: Intermediate state to introduce a delay. Transitions to STALL\_2.

STALL\_2: Another intermediate state to introduce a delay. Transitions to UPDATE\_AGES.

UPDATE\_AGES: Updates the age of the cache lines. Resets the age of the hit set and increments the ages of other sets. Transitions to IDLE.

CHECK\_STATUS: Checks the hit\_miss signal.

If hit\_miss is high, transition to UPDATE\_AGES.

If hit\_miss is low, transition to STALL\_1.

Functionality:

The module uses a state machine to manage cache operations and age updates.

In the UPDATE\_AGES state, it identifies the hit set and updates the ages accordingly.

control\_unit\_tb (Testbench)

Purpose:

The testbench verifies the functionality of the control\_unit module by simulating clock and reset signals, generating random opcodes, hit/miss signals, hit/miss sets, and ages.

Signals:

clk: Clock signal (toggled every half period).

rst\_b: Active low reset signal (asserted for a defined pulse duration and then deasserted).

opcode: Randomly generated operation code (read/write).

hit\_miss: Randomly generated hit/miss signal.

hit\_miss\_set: Randomly generated set index for hit/miss detection.

ages: Randomly generated age information for cache blocks.

**Functionality**:

Clock Generation: Toggles the clk signal for a specified number of cycles.

Reset Signal: Asserts rst\_b low initially and then deasserts it after a specified pulse duration.

Random Signal Generation: Randomly generates values for opcode, hit\_miss, hit\_miss\_set, and ages to simulate various scenarios and test the control\_unit.

**Summary**

The control\_unit module is a finite state machine that manages cache read/write operations, updates cache line ages, and transitions between states based on inputs like opcode and hit\_miss. The testbench simulates different scenarios by generating random input signals to verify the correct functionality of the control\_unit. This ensures that the cache control logic operates as expected under various conditions.



