Dupli Checker

PLAGIARISM SCAN REPORT

0% Plagiarised	100% Unique	Date	2022-04-10		
		Words	347		
		Characters	3069		

Content Checked For Plagiarism

```
module cache_controller (
        clock,
        reset_n,

    // Declaring CPU signals
        cpu_req_addr,
        cpu_req_datain,
        cpu_req_dataout,
        cpu_req_rw,
        cpu_req_valid,

    // Declaring Cache ready signal
        cache_ready,

    // Declaring Main memory signals
        mem_req_addr,
        mem_req_datain,
        mem_req_dataout,
        mem_req_rw,
        mem_req_valid,
        mem_req_ready
);

// Declaring the FSM states
parameter IDLE        = 2'b00;
parameter COMPARE_TAG = 2'b01;
parameter ALLOCATE    = 2'b10;
parameter WRITE_BACK  = 2'b11;

// Initial inputs
input clock;
input reset_n;

// Declaring the inputs and outputs for CPU requests to cache controller
input [31:0] cpu_req_addr;
input [127:0] cpu_req_datain;
input cpu_req_rw; // For cpu writes, the value is 1 & for cpu reads, its 0
input cpu_req_valid;

output [31:0] cpu_req_dataout;

// Declraing the inputs and outputs for Main Memory Requests from cache controller
input [127:0] mem_req_datain;
input mem_req_ready;

output [31:0] mem_req_addr;
output [31:0] mem_req_dataout;
output mem_req_rw;
output mem_req_valid;

// For cache ready
output cache_ready;

// Cache Memory = Tag Memory + Data Memory

// Tag Memory = tag + valid bit + dirty bit
reg [19:0] tag_mem [1023:0];

// Data part of the cache
reg [127:0] data_mem [1023:0];

// Declaring the necessary temporary variables as wire
wire [17:0] cpu_addr_tag;
wire [9:0] cpu_addr_index;
wire [1:0] cpu_addr_blk_offset;
wire [1:0] cpu_addr_byte_offset;
wire [19:0] tag_mem_entry;
wire [127:0] data_mem_entry;
wire hit;

// Declaring the required variables as register
reg [1:0] present_state, next_state;
reg [31:0] cpu_req_dataout, next_cpu_req_dataout;
reg [31:0] cache_read_data;
reg cache_ready, next_cache_ready;
reg [31:0] mem_req_addr, next_mem_req_addr;
reg mem_req_rw, next_mem_req_rw;
reg mem_req_valid, next_mem_req_valid;
reg [127:0] mem_req_dataout, next_mem_req_dataout;

reg write_datamem_mem; // Write operation from Main Memory
reg write_datamem_cpu; // Write operation from CPU
reg tagmem_enable;
reg valid_bit, dirty_bit;

reg [31:0] cpu_req_addr_reg, next_cpu_req_addr_reg;
reg [127:0] cpu_req_datain_reg, next_cpu_req_datain_reg;
reg cpu_req_rw_reg, next_cpu_req_rw_reg;

// Defining the range of various parts of CPU Address
// CPU Address = tag + index + block offset + byte offset
```

```
assign cpu_addr_tag = cpu_req_addr_reg[31:14];
assign cpu_addr_index = cpu_req_addr_reg[13:4];
assign cpu_addr_blk_offset = cpu_req_addr_reg[3:2];
assign cpu_addr_byte_offset = cpu_req_addr_reg[1:0];

assign tag_mem_entry = tag_mem[cpu_addr_index];
assign data_mem_entry = data_mem[cpu_addr_index];
assign hit = tag_mem_entry[19] && (cpu_addr_tag == tag_mem_entry[17:0]);

// Loading initial values for Data memory and Tag memory
initial begin
$readmemh("data_memory.mem", data_mem);
end

initial begin
$readmemh("tag_memory.mem", tag_mem);
end

always @ (posedge clock or negedge reset_n)
begin
  if(!reset_n)
  begin
      tag_mem[cpu_addr_index]
```

Matched Source

No plagiarism found