CA_Lab6

K. Ram Mohan

COE19B055

Verilog Code:

```
module CACHE (read_addr, read_data, read_enable, write_addr, write_data, write_enable,
clk);
   input[20:0] read_addr, write_addr;
   input read_enable, write_enable;
   input clk;
   input[63:0] write_data;


   output reg[63:0] read_data;



   // W=4(Since 16 Words. Our memory is word addressable). Since we took 21-bit
instruction address
   // 2^17/2^10 = 2^7
   reg[63:0] main_mem[0:131071][0:15];


   reg[63:0] cache[0:1023][15:0];
   // 21-10-4 = 7
   reg[6:0] tags[0:1023];


   wire [6:0] read_tag;
   wire [9:0] read_line;
   wire [3:0] read_offset;


   wire [6:0] write_tag;
   wire [9:0] write_line;
   wire [3:0] write_offset;
```

```
integer i, j;
assign read_tag = read_addr[20:14];
assign read_line = read_addr[13:4];
assign read_offset = read_addr[3:0];

assign write_tag = write_addr[20:14];
assign write_line = write_addr[13:4];
assign write_offset = write_addr[3:0];

// wire val[1023:0], dirty[1023:0];
reg val[1023:0], dirty[1023:0];

initial begin
   for (i = 0; i < 131072; i = i + 1) begin
      for (j = 0; j < 16; j = j + 1) begin
         main_mem[i][j] = 0;
      end
   end

   // Initial Invalid Tags
   for (i = 0; i < 1024; i = i + 1) begin
      val[i] = 0;
      dirty[i] = 0;
   end
end

always @(posedge clk) begin
   if (read_enable) begin
```

```
    if (tags[read_line] == read_tag && val[read_line] == 1'b1) begin
        $monitor("%d Cache Hit : Reading Value.", $time);
        read_data = cache[read_line][read_offset];
    end
    else begin
        $display("%d Could not read. Miss", $time);
        // check if line is DIRTY
        if (dirty[write_line] == 1'b1) begin
            // write cache line to memory block (cache line is modified)
            $display("%d Encountered a dirty line.", $time);
            for(i=0; i < 16; i = i+1) begin
                main_mem[{tags[read_line], read_line}][i] = cache[read_line][i];
            end
            dirty[read_line] = 1'b0;
        end
        else begin
            // bring correct block to cache
            $display("%d Line is not dirty, storing in cache...", $time);
            for(i=0; i < 16; i = i+1)begin
                cache[read_line][i] = main_mem[{read_tag, read_line}][i];
            end
            val[read_line] = 1'b1;
            tags[read_line] = read_tag;
        end
    end
end

if (write_enable) begin
    // check if block is in cache and is val
```

```
    if (tags[write_line] == write_tag && val[write_line] == 1'b1) begin
        $display("%d Cache Hit : Writing Value.", $time);
        cache[write_line][write_offset] = write_data;
        dirty[write_line] = 1'b1;
    end

    // replace with correct line
    else begin
        $display("%d Could not write. Miss", $time);
        // check if line is dirty
        if (dirty[write_line] == 1'b1) begin
            // write cache line to memory block
            $display("%d # Encountered a dirty line.", $time);
            for(i=0; i < 16; i = i+1)begin
                main_mem[{tags[write_line], write_line}][i] = cache[write_line][i];
            end
            dirty[write_line] = 1'b0;
        end
        else begin
            // bring correct block to cache
            $display("%d  Line is not dirty, storing in cache...", $time);
            for(i=0; i < 16; i = i+1) begin
                cache[write_line][i] = main_mem[{write_tag, write_line}][i];
            end
            val[write_line] = 1'b1;
            tags[write_line] = write_tag;
        end
    end
end
```

```
    end
endmodule

module CACHE_TB;
    reg[20:0] read_addr, write_addr;
    reg[63:0] write_data;
    reg clk, read_enable, write_enable;

    wire[63:0] read_data;

    CACHE cache(read_addr, read_data, read_enable, write_addr, write_data, write_enable,
clk);

    initial begin
        clk = 0;
        forever begin
            #2 clk = ~clk;
        end
    end


    initial begin
        //Read Miss
#10 read_enable = 1'b1;
read_addr = 21'b1000000_1110000000_1011;
#10 read_enable = 1'b0;
#10$display("%d # %b: %h", $time, read_addr, read_data);

        //Write Hit
#10 write_enable = 1'b1;
```

```
write_addr = 21'b1000000_1110000000_1011;

write_data = 64'habcdef1010110110;

#10 write_enable = 1'b0;


//Read Hit

#10 read_enable = 1'b1;

read_addr = 21'b1000000_1110000000_1011;

#10 read_enable = 1'b0;

#10 $display("%d # %b: %h", $time, read_addr, read_data);


//Read Miss

    // Since Different Tag(Which means that the modulo of frame is same but came from different division of main memory). So miss will happen. But the same line is already written previously. so data present in that line will be written to main memory.

#10 read_enable = 1'b1;

read_addr = 21'b1110000_1110000000_1011;

#10 read_enable = 1'b0;

#10 $display("%d # %b: %h", $time, read_addr, read_data);


//Read Miss

#10 read_enable = 1'b1;

read_addr = 21'b1000000_1110000000_1011;

#10 read_enable = 1'b0;

#10 $display("%d # %b: %h", $time, read_addr, read_data);


//Write Miss

#10 write_enable = 1'b1;

write_addr = 21'b1010000_1110000000_1011;

write_data = 64'h01011120abcdef;

#10 write_enable = 1'b0;
```

```
//Read Hit
#10 read_enable = 1'b1;
read_addr = 21'b1010000_1110000000_1011;
#10 read_enable = 1'b0;
#10 $display("%d # %b: %h", $time, read_addr, read_data);

#1000 $finish;
   end

endmodule
```

Output: