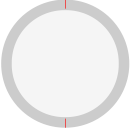
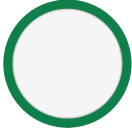


PLAGIARISM SCAN REPORT

	0% Plagiarised		100% Unique	Date	2022-04-10
				Words	406
				Characters	3137

Content Checked For Plagiarism

```
// Defining the testbench module
module cache_controller_tb;

reg clock;
reg reset_n;

// Declaring the required variables as register
reg [31:0] cpu_req_addr;
reg [127:0] cpu_req_datain;
reg cpu_req_rw; // 0 for reads and 1 for writes
reg cpu_req_valid;
reg [127:0] mem_req_datain;
reg mem_req_ready;

// Declaring the temporary variables as wires
wire [31:0] cpu_req_dataout;
wire [31:0] mem_req_addr;
wire [127:0] mem_req_dataout;
wire mem_req_rw;
wire mem_req_valid;

// Defining the Design under test(dut)

cache_controller dut (
    .clock      (clock),
    .reset_n    (reset_n),

    // For all the CPU signals
    .cpu_req_addr  (cpu_req_addr),
    .cpu_req_datain (cpu_req_datain),
    .cpu_req_dataout (cpu_req_dataout),
    .cpu_req_rw    (cpu_req_rw),
    .cpu_req_valid (cpu_req_valid),
    .cache_ready   (cache_ready),

    // For all the Memory signals
    .mem_req_addr  (mem_req_addr),
```

```

.mem_req_datain  (mem_req_datain),
.mem_req_dataout (mem_req_dataout),
.mem_req_rw      (mem_req_rw),
.mem_req_valid   (mem_req_valid),
.mem_req_ready   (mem_req_ready)
);

always #5 clock = ~clock; // Changing the clock at an delay of 5

// Defining the function for reading from the CPU
task read_cpu (input [31:0]addr);
begin
    #20 cpu_req_addr = addr; cpu_req_rw = 1'b0; cpu_req_valid = 1'b1; // Execute the line at a delay of 20
    #10 cpu_req_addr = 32'd0; cpu_req_rw = 1'b0; cpu_req_valid = 1'b0; // Execute the line at a delay of 10
end
endtask

// Defining the function for writing by the CPU
task write_cpu (input [31:0]addr, input [127:0]data);
begin
    #20 cpu_req_addr = addr; cpu_req_rw = 1'b1; cpu_req_valid = 1'b1; cpu_req_datain = data; // Write to the Cache i.e. AB at
a delay of 20
    #10 cpu_req_addr = 32'd0; cpu_req_rw = 1'b0; cpu_req_valid = 1'b0; cpu_req_datain = 128'd0; // Execute the line at a
delay of 10
end
endtask

// Defining the function for resetting the values
task reset_values ();
begin
    clock = 1'b1; reset_n = 1'b0;

    // For CPU signals
    cpu_req_addr = 32'd0;
    cpu_req_datain = 128'd0;
    cpu_req_rw = 1'b0;
    cpu_req_valid = 1'b0;

    // For Memory Signals
    mem_req_datain = 128'd0;
    mem_req_ready = 1'b1;

    #20 reset_n = 1'b1; // Execute this line at a delay of 20
end
endtask

initial begin
    reset_values();
    write_cpu (332'hAB00,128'h1122); // Write signal by the CPU

    read_cpu (32'hBB00);           // We have a read miss if clean (same tag, diff index)
    read_cpu (32'hAB00);           // We have a read hit (same tag, same index)

    @(negedge mem_req_valid) mem_req_ready = 1'b0;

```

```
#20 mem_req_datain = 128'h3344; mem_req_ready = 1'b1; // At a delay of 20

read_cpu (32'hEB00);          //We have a read miss if dirty (diff tag, same index)
@(negedge mem_req_valid) mem_req_ready = 1'b0;
#20 mem_req_ready = 1'b1; // At a delay of 20

@(negedge mem_req_valid) mem_req_ready = 1'b0;
#30 mem_req_datain = 128'h5566; mem_req_ready = 1'b1; // At a delay of 30
end
endmodule
```

Matched Source

No plagiarism found

Check By:  Dupli Checker