```
`timescale 1ns / 1ps

module slow_division_tb;

  // Parameters
  parameter CLK_PERIOD = 10; // Clock period in nanoseconds

  // Inputs
  reg clk;
  reg [7:0] dividend;
  reg [3:0] divisor;

  // Outputs
  wire [7:0] quotient;
  wire [3:0] remainder;

  // Instantiate the slow division module
  slow_division slow_div_inst(
    .clk(clk),
    .dividend(dividend),
    .divisor(divisor),
    .quotient(quotient),
    .remainder(remainder)
  );

  // Clock generation
  always #((CLK_PERIOD / 2)) clk = ~clk;

  // Stimulus
  initial begin
    $dumpfile("slow_division_tb.vcd");
    $dumpvars(0, slow_division_tb);

    clk = 0;
    #10;

    // Test case 1
    #10;
    dividend = 32;
    divisor = 5;
    #10;
    $display("Test Case 1:");
    $display("Dividend = %d, Divisor = %d", dividend, divisor);
    #10;
```

```
    // Perform division
    // Compare the quotient and remainder with expected values
    if (quotient === 6 && remainder === 2)
      $display("Test Case 1 Passed");
    else
      $display("Test Case 1 Failed");

    // Test case 2
    #10;
    dividend = 73;
    divisor = 9;
    #10;
    $display("Test Case 2:");
    $display("Dividend = %d, Divisor = %d", dividend, divisor);
    #10;
    // Perform division
    // Compare the quotient and remainder with expected values
    if (quotient === 8 && remainder === 1)
      $display("Test Case 2 Passed");
    else
      $display("Test Case 2 Failed");

    // Repeat for other test cases

    // End simulation
    $finish;
  end

endmodule
```