```
// Eugene Ngo
// 1/20/2023
// EE 371
// Lab 2 Task 1

// DE1_SoC is the top-level module that defines the I/Os for the DE-1 SoC board.
// DE1_SoC uses switches SW 3-0 to provide input data for the RAM and switches SW 8-4 to specify the write address
// for the RAM module. It uses SW9 as the Write signal. Using hexadecimal values, it shows the write address value
// on the 7-segment displays HEX5-4, the read address value on HEX3-2, the write data on HEX1, and the read data
// on HEX0.

module DE1_SoC_task2 (HEX0, HEX1, HEX2, HEX3, HEX4, HEX5, KEY, SW, CLOCK_50);
   output logic [6:0]  HEX0, HEX1, HEX2, HEX3, HEX4, HEX5;
   input  logic [3:0]  KEY;
   input  logic [9:0]  SW;
   input logic CLOCK_50;

   // Assigning intermediate logic to DE1_SoC peripherals to improve program readability
   logic [4:0] addr_r, addr_w;
   logic [3:0] data_in, data_out;
   logic write, reset;
   assign addr_w = SW[8:4];
   assign data_in = SW[3:0];
   assign write = ~KEY[3];
   assign reset = ~KEY[0];

   // RAM instantiates the dual-port memory module previously made. It connects to the corresponding intermediate
   // logics specified above, and it uses CLOCK_50 as the clock signal.
   ram32x4 RAM (.clock(CLOCK_50), .data(data_in), .rdaddress(addr_r), .wraddress(addr_w), .wren(write), .q(data_out));

   // c1 increments addr_r once every second. It uses CLOCK_50 as the clock signal.
   counter c1 (.addr_r, .clk(CLOCK_50), .reset);

   // Smaller 2-clock-cycle counter for testbench purpose.
   // counter #(1) c1 (.addr_r, .clk(CLOCK_50), .reset);

   // hd1 takes addr_r, addr_w, data_in, and data_out, and displays those values in hexadecimal to the corresponding
   // hex display.
   seg7 hd1 (.addr_r, .addr_w, .data_in, .data_out, .HEX5, .HEX4, .HEX3, .HEX2, .HEX1, .HEX0);

endmodule

// DE1_SoC_task2_testbench tests all expected, unexpected, and edgecase behaviors
`timescale 1 ps / 1 ps
module DE1_SoC_task2_testbench();
   logic [6:0]  HEX0, HEX1, HEX2, HEX3, HEX4, HEX5;
   logic [3:0]  KEY;
   logic [9:0]  SW;
   logic CLOCK_50;

   DE1_SoC_task2 dut (.HEX0, .HEX1, .HEX2, .HEX3, .HEX4, .HEX5, .KEY, .SW, .CLOCK_50);

   // Setting up a simulated clock.
   parameter CLOCK_PERIOD = 100;
   initial begin
      CLOCK_50 <= 0;
      forever #(CLOCK_PERIOD/2) CLOCK_50 <= ~CLOCK_50; // Forever toggle the clock
   end

   // Setting up intermediate logic to improve readability
   logic [4:0] addr_w;
   logic [3:0] data_in;
   logic write, reset;
   assign SW[8:4] = addr_w;
   assign SW[3:0] = data_in;
   assign KEY[3] = ~write;
   assign KEY[0] = ~reset;


   // Trying all combinations of inputs (x and y).
   initial begin
      reset <= 1;                                                                 repeat(3) @(posedge CLOCK_50);

      // updating address 5-9 while reading address 0-4
      reset <= 0; addr_w <= 5'b00101; data_in <= 4'b1111; write <= 1'b1; repeat(2) @(posedge CLOCK_50);
                  addr_w <= 5'b00110; data_in <= 4'b1111; write <= 1'b1; repeat(2) @(posedge CLOCK_50);
                  addr_w <= 5'b00111; data_in <= 4'b1111; write <= 1'b1; repeat(2) @(posedge CLOCK_50);
                  addr_w <= 5'b01000; data_in <= 4'b1111; write <= 1'b1; repeat(2) @(posedge CLOCK_50);
                  addr_w <= 5'b01001; data_in <= 4'b1111; write <= 1'b1; repeat(2) @(posedge CLOCK_50);

      // reading address 5-9 and reading unmodified address 10-14
                                                          write <= 1'b0; repeat(20) @(posedge CLOCK_50);

      $stop;
   end
endmodule
```