Josh Monson

ECEN 620 Assignment 2.1

9/18/14

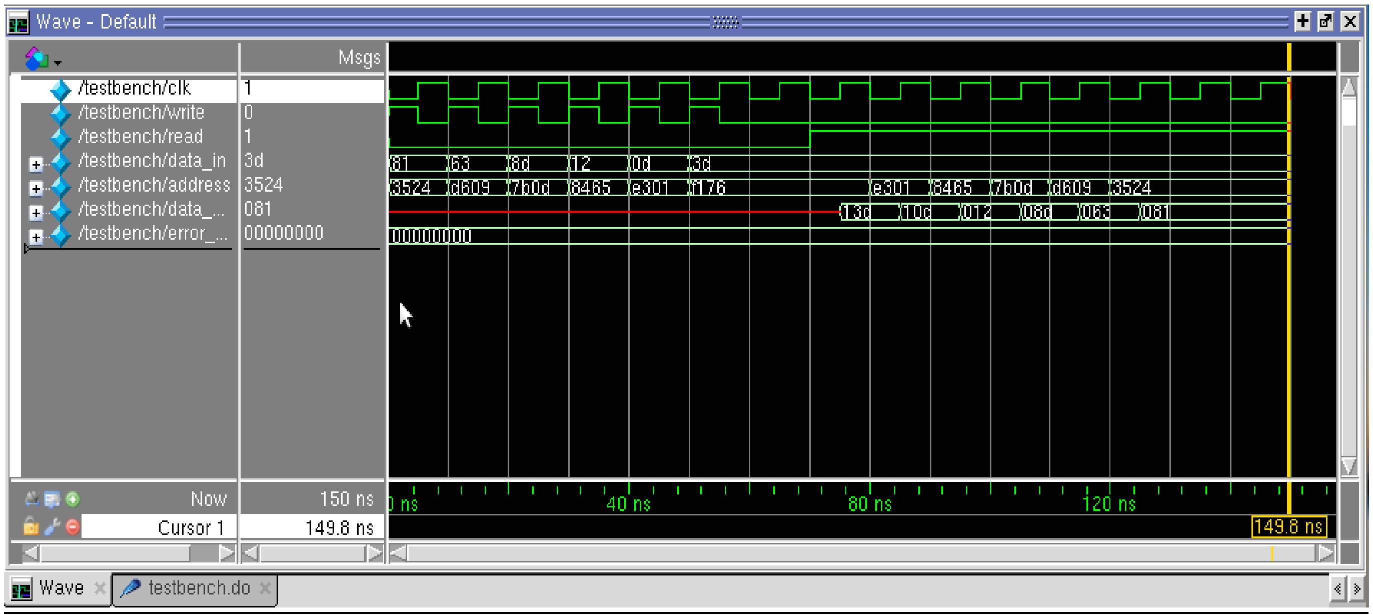
Memory Model Code

```
`default_nettype none
  module my_mem(clk,
             write,
             read,
             data_in,
             address,
             data_out);

input  logic clk;
input  logic write;
input  logic read;
input  logic [7:0] data_in;
input  logic [15:0] address;
output logic [8:0] data_out;

   // Declare a 9-bit associative array using the logic data type
   logic [8:0] mem_array[shortint];

   always @(posedge clk) begin
      if (write)
        mem_array[address] = {^data_in, data_in};
      else if (read)
        data_out =  mem_array[address];
   end

endmodule
```

Test Bench Code

```
`default_nettype none
`timescale 1ns/100ps

module testbench ();

  logic clk;
  logic write;
  logic read;
  logic [7:0] data_in;
  logic [15:0] address;
  logic [8:0] data_out;

  shortint address_array[];
  byte data_to_write_array[];
  bit [8:0] data_read_expect_assoc[shortint];
  int error_counter;
  bit [8:0] data_read_queue[$];

  initial begin
    clk = 0;
    error_counter = 0;
    read = 0;
    address_array = new[6];
    data_to_write_array = new[6];

    foreach(address_array[i]) begin
      address_array[i] = $random;
      data_to_write_array[i] = $random;
      data_read_expect_assoc[address_array[i]] =  {^data_to_write_array[i],
data_to_write_array[i]};
    end

    foreach(address_array[i]) begin
      write_to_memory(address_array[i], data_to_write_array[i]);
    end

    address_array.reverse();

    foreach(address_array[i]) begin
      @(negedge clk)
      read = 1;
      address = address_array[i];
      @(posedge clk)
      #1
      data_read_queue.push_back(data_out);
      if(data_read_expect_assoc[address] != data_out) begin
        $display("Found Error: %03X %03X", data_read_expect_assoc[address],
data_out);
        error_counter = error_counter + 1;
      end
    end

   $display("Print Read Values");
   foreach(data_read_queue[i]) begin
    $display("\t%03X", data_read_queue[i]);
   end

   $display("Error Count: %d", error_counter);

  end

  task write_to_memory(input shortint addr, byte data);
    write = 1;
    data_in = data;
    address = addr;
    @(posedge clk);
     write = 0;
    @(negedge clk);
  endtask;
  always begin
    #5 clk = ~clk;
  end
  my_mem mem(clk,
           write,
           read,
           data_in,
           address,
           data_out);
   endmodule
```

Screen Shot of Waveform



Working Transcript Window

```
vsim -novopt testbench

# Refreshing /net/fpga1/users/joshuas2/ECEn620/Assignments/Assignment2.1/work.testbench
# Loading sv_std.std
# Loading work.testbench
# Refreshing /net/fpga1/users/joshuas2/ECEn620/Assignments/Assignment2.1/work.my_mem
# Loading work.my_mem

# Print Read Values
#          13d
#          10d
#          012
#          08d
#          063
#          081

# Error Count:         0
```

Non-Working Transcript Window

```
# vsim -novopt testbench
# Refreshing /net/fpga1/users/joshuas2/ECEn620/Assignments/Assignment2.1/work.testbench
# Loading sv_std.std
# Loading work.testbench
# Refreshing /net/fpga1/users/joshuas2/ECEn620/Assignments/Assignment2.1/work.my_mem
# Loading work.my_mem
# Found Error: 13d 000
# Found Error: 10d 000
# Found Error: 012 000
# Found Error: 08d 000
# Found Error: 063 000
# Found Error: 081 000

# Print Read Values

#           000
#           000
#           000
#           000
#           000
#           000

# Error Count:          6
```