Joshua Monson
Assignment 4.1
10/3/2014

Overview

In this assignment I have added an interface to the memory model from the previous assignment. You'll notice that in addition to adding the protocol function and checker (read and write checker) and *parity* function I have also added *mread()* and *mwrite()* functions to the interface. These functions set the read/write, data_in, and address signals. I wasn't sure if it was bad practice to have the interface advance time. It seems like that should be handled in the test bench. In this report, you will find the system Verilog code for the test bench, memory model, and memory interface. Additionally, I have included copies of the console transcripts to show the testbench works and can catch specific errors. The console transcripts show two types of errors: "Incorrect Read Errors" and "Read/Write Signal Errors." Incorrect Read Errors occur when the data read from the memory is not what was expected. Read/Write Signals error occur when both the read and write signal on the memory interface are asserted at the same time. I have also included a waveform showing the memory I/O and error counters. Incorrect Read Errors are tallied by the signal *ErrorCount* (see waveform). Read/Write Signals errors are tallied by *rdwr_error_count* signal (see waveform).

TestBench Code

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

module testbench ();

  shortint address;
  byte data;

  //Clock Generator
  bit clk = 0;
  always #5ns clk = ~clk;

  //Memory Interface
  mem_if mem_bus(clk);

  typedef struct {
      shortint address;
      byte data_to_write;
      byte expected_read;
```

```
    byte actual_read;
} ScoreBoardEntry;

ScoreBoardEntry ScoreBoard[];

int ErrorCount = 0;

///////////////////////////
//    Test Bench Code    //
///////////////////////////

initial begin

  //Initialize Score Board
  ScoreBoard = new[6];

  //Generate Random Writes
  foreach(ScoreBoard[i]) begin
    address = $random;
    data = $random;
    ScoreBoard[i].address = address;
    ScoreBoard[i].data_to_write = data;
    ScoreBoard[i].expected_read = {^data, data};
  end

  //Test Protocol Error Checker
  //This should produce three Errors
  mem_bus.mwrite(0, 0);
  mem_bus.mread(0);

  @(negedge mem_bus.clk);
  @(negedge mem_bus.clk);
  @(negedge mem_bus.clk);

  mem_bus.write = 0;
  mem_bus.read = 0;

  //Perform Writes
  foreach(ScoreBoard[i]) begin
    write_memory(ScoreBoard[i]);
  end
  @(negedge mem_bus.clk);
  mem_bus.write = 0;

  //Rearrange Array for Reads
  ScoreBoard.shuffle();

  //Perform Reads and Check Results
  foreach(ScoreBoard[i]) begin
   read_memory(ScoreBoard[i]);
   check(ScoreBoard[i]);
  end
  mem_bus.read = 0;

  //Display the Actual Values Read
  $display("Print Read Values");
  foreach(ScoreBoard[i]) begin
```

```
     $display("\t%03X", ScoreBoard[i].actual_read);
    end

    $display("Tests Performed: %d", ScoreBoard.size());
    $display("Error Count: %d", ErrorCount);
    $display("Read/Write Errors: %d", mem_bus.rdwr_error_count);


  end

  task automatic write_memory(ref ScoreBoardEntry Entry);
    @(negedge mem_bus.clk);
    mem_bus.mwrite(Entry.address, Entry.data_to_write);
  endtask

  task automatic read_memory(ref ScoreBoardEntry Entry);
    @(negedge mem_bus.clk);
    mem_bus.mread(Entry.address);
    @(negedge mem_bus.clk);
    Entry.actual_read = mem_bus.data_out;
  endtask

  function check(ScoreBoardEntry Entry);
    if( Entry.expected_read != Entry.actual_read) begin
      $display("Found Error: %03X %03X", Entry.expected_read,
Entry.actual_read);
      ErrorCount++;
    end
  endfunction

  my_mem mem(mem_bus);

endmodule
```

Memory Model and Interface

```
`default_nettype none

interface mem_if(input bit clk);
   logic write;
   logic read;
   logic [7:0] data_in;
   logic [15:0] address;
   logic [8:0] data_out;

   ///////////////////////////
   //     Read Function     //
   ///////////////////////////
   //Note: does not return read value
   function void mread(logic [15:0] addr);
    read = 1;
    address = addr;
   endfunction
   ///////////////////////////
   //    Write Function     //
   ///////////////////////////
   function void mwrite(logic [15:0] addr, logic [7:0] data);
    write = 1;
    data_in = data;
    address = addr;
   endfunction

   ///////////////////////////
   //Test Read Write Checker//
   ///////////////////////////
   always @(posedge clk) ReadWriteChecker(read, write);

   ///////////////////////////
   //   Read Write Checker  //
   ///////////////////////////
   int rdwr_error_count;
   function  logic ReadWriteChecker(logic rd, logic wr);
     if(rd == 1 && wr == 1) begin
       $display("Error@%d: Read and Write asserted at the same time\n",
$time);
       rdwr_error_count++;
     end
   endfunction

   ///////////////////////////
   //    Even Parity Calc.  //
   ///////////////////////////
   function logic parity(logic [7:0] din);
     return (^din);
   endfunction

endinterface


module my_mem(mem_if mem_bus);

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

   always @(posedge mem_bus.clk) begin
      //$display("Clk Event mem_bus.write=%d time: %d", mem_bus.write,
$time);
      if (mem_bus.write) begin
        //$display("Writing: %04x to %04x\n", {ev_parity(mem_bus.data_in),
mem_bus.data_in}, mem_bus.address);
        mem_array[mem_bus.address] = {mem_bus.parity(mem_bus.data_in),
mem_bus.data_in};
      end else if (mem_bus.read)
        mem_bus.data_out =  mem_array[mem_bus.address];
   end

endmodule
```

Console Transcript: Zero Memory Read Errors

```
do testbench.do

# vsim -novopt testbench
# Refreshing
/net/fpga1/users/joshuas2/ECEn620/Assignments/Assignment4.1/work.testbenc
h
# Loading sv_std.std
# Loading work.testbench
# Refreshing
/net/fpga1/users/joshuas2/ECEn620/Assignments/Assignment4.1/work.mem_if
# Loading work.mem_if
# Refreshing
/net/fpga1/users/joshuas2/ECEn620/Assignments/Assignment4.1/work.my_mem
# Loading work.my_mem
# Print Read Values

#       063
#       08d
#       081
#       03d
#       00d
#       012

# Tests Performed:          6
# Incorrect Read Errors:          0
# Read/Write Signal Errors:          0
```

Console Transcript: Memory Read Errors

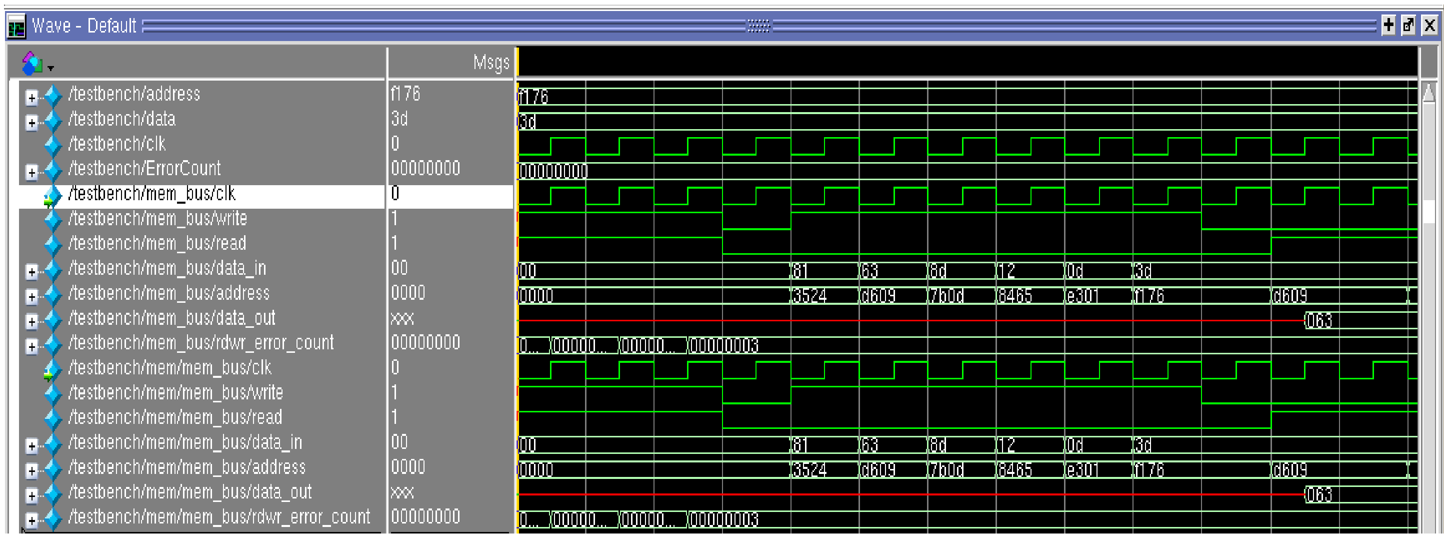
```
# vsim -novopt testbench

# Refreshing
/net/fpga1/users/joshuas2/ECEn620/Assignments/Assignment4.1/work.testbenc
h
```

```
# Loading sv_std.std
# Loading work.testbench
# Refreshing
/net/fpga1/users/joshuas2/ECEn620/Assignments/Assignment4.1/work.mem_if
# Loading work.mem_if
# Refreshing
/net/fpga1/users/joshuas2/ECEn620/Assignments/Assignment4.1/work.my_mem
# Loading work.my_mem

# Found Error: 063 000
# Found Error: 08d 000
# Found Error: 081 000
# Found Error: 03d 000
# Found Error: 00d 000
# Found Error: 012 000

# Print Read Values

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

# Tests Performed:          6
# Incorrect Read Errors:          6
# Read/Write Signal Errors:          0
```

Console Transcript: Read Write Signal Errors

```
do testbench.do

# Refreshing
/net/fpga1/users/joshuas2/ECEn620/Assignments/Assignment4.1/work.testbenc
h

# Loading sv_std.std
# Loading work.testbench
# Refreshing
/net/fpga1/users/joshuas2/ECEn620/Assignments/Assignment4.1/work.mem_if

# Loading work.mem_if

# Refreshing
/net/fpga1/users/joshuas2/ECEn620/Assignments/Assignment4.1/work.my_mem

# Loading work.my_mem
# Error@                 50: Read and Write asserted at the same time
# # Error@                150: Read and Write asserted at the same time

# Error@                250: Read and Write asserted at the same time
# Print Read Values

#        063
#        08d
#        081
#        03d
#        00d
#        012

# Tests Performed:          6
# Incorrect Read Errors:          0
# Read/Write Signal Errors:          3
```

Waveform: Showing I/0 and Error Counters



```
/testbench/address                    f176
/testbench/data                       3d
/testbench/clk                        0
/testbench/ErrorCount                 00000000
/testbench/mem_bus/clk                0
/testbench/mem_bus/write              1
/testbench/mem_bus/read               1
/testbench/mem_bus/data_in            00
/testbench/mem_bus/address            0000
/testbench/mem_bus/data_out           xxx
/testbench/mem_bus/rdwr_error_count   00000000
/testbench/mem/mem_bus/clk            0
/testbench/mem/mem_bus/write          1
/testbench/mem/mem_bus/read           1
/testbench/mem/mem_bus/data_in        00
/testbench/mem/mem_bus/address        0000
/testbench/mem/mem_bus/data_out       xxx
/testbench/mem/mem_bus/rdwr_error_count 00000000
```