**Chapter 2, Data Types homework 1**

The purpose of this homework is to:

1. Practice creating dynamic arrays, associative arrays, and queues
2. Practice using $random
3. Practice writing an easy to expand test-bench

This HW will test a synchronous 8-bit x64K (512kBit) RAM. The RAM will read on the positive edge of the clock when input read =1 and write on the positive edge of the clock when input write = 1. Even parity will be calculated on data written to the RAM and placed in the 9th bit of the memory. The partially completed memory model is below.

`default\_nettype none

module my\_mem(

input var logic clk,

input var logic write,

input var logic read,

input var logic [7:0] data\_in,

input var logic [15:0] address,

output logic [8:0] data\_out

);

// Declare a 9-bit associative array using the logic data type

<Put your declaration here>

always @(posedge clk) begin

if (write)

mem\_array[address] = {^data\_in, data\_in};

else if (read)

data\_out = mem\_array[address];

end

endmodule

**Your testbench will:**

1. Perform 6 writes of random data to random addresses followed by 6 reads to the same addresses in reverse order. To do this:
   1. Create a dynamic array of 6 random addresses to write called *address\_array*
   2. Create a dynamic array of 6 random bytes to write called *data\_to\_write\_array*
   3. Create an associative array of data expected to be read indexed by the address read from called *data\_read\_expect\_assoc*
   4. Make your test-bench self checking by comparing the data read with the *data\_read\_expect\_assoc* array. Maintain an error counter
   5. Create a queue of data read called *data\_read\_queue*
2. At the end of the test traverse the *data\_read\_queue*, print out the data read, and print out the error counter.

Deliverables:

1. Code for completed memory model and test-bench
2. Waveform showing at a minimum the I/O of the memory model and the error counter.
3. Copy of transcript window showing the print out of *data\_read\_queue* and the error counter equaling 0.