**Chapter 4, Connecting the Testbench and Design homework 2**

The purpose of this homework is to practice using clocking blocks, programs, and assertions to improve your solution for Chapter 4, Connecting the Testbench and Design homework 1. Expand on this solution by:

1. Add a clocking block to your interface for all of the I/O of module my\_mem and use this clocking block to drive/observe these signals in your testbench. The testbench should only drive/observe signals to/from the DUT through the clocking block. The clock domain for your clocking block is posedge clk since the DUT’s flip-flops are positive edge triggered.
2. Separate your testbench into a program block that provides stimulus and checks results and a top level that instantiates the DUT, interface, and program similar to Figure 1. Your top level should also create the clock.



Figure 1: Testbench hierarchy

1. Ensure that your testbench will work for both 0-delay my\_mem model as well a non-0 delay my\_mem model by replacing the following line in my\_mem.sv

mem\_bus.data\_out = mem\_array[mem\_bus.address]; // 0-delay

with

mem\_bus.data\_out = #75ns mem\_array[mem\_bus.address]; // non-0 delay

1. Modify your checker that watches for both write =1 and read =1 to use an assertion. Continue to test this in your testbench.
2. The clock created in your testbench shall be 10MHz.
3. Your testbench should only perform the minimum number of reads necessary (i.e. 6).

Deliverables:

1. Code for the interface, module top, and program test.
2. Waveform showing at a minimum the I/O of the memory model, error counter, and assertion for a 0 delay my\_mem model and transcript file reporting 0 memory read errors and at least 1 assertion failure.
3. Waveform showing at a minimum the I/O of the memory model, error counter, and assertion for a non-0 delay my\_mem model and transcript file reporting 0 memory read errors and at least 1 assertion failure