ECE 571
Spring 2021
Homework 6

1. Modify your (if it was working) or my Verilog code for the garage door assignment to make use of all appropriate SystemVerilog constructs and idioms (at a minimum use the **always_comb** and **always_ff** procedural blocks and enumerated types for states). Confirm that it is working with my testbench before submitting.

2. Modify your (if it was working) or my testbench for the pipelined multiplier to read test cases from a text file. The file will consist of lines containing a multiplier and multiplicand (separated by one or more spaces). The filename will be provided on the **vsim** command line using **+FILENAME=** and retrieved using **$value$plusargs**. You'll find details on using **$value$plusargs** and specifying runtime values on the **vsim** command line in the Using Mentor Questa at PSU document you've previously read (found in D2L Content -> Resources -> Using Mentor Questa at PSU). Use an additional command line argument **+RADIX=** to set the radix to either decimal or hexadecimal (**+RADIX=d** or +**RADIX=h**).

Using the code for the SimpleBus processor/memory interface from the Thomas book, Section 6.3 or similar code provided by the instructor, make the following modifications and extensions.

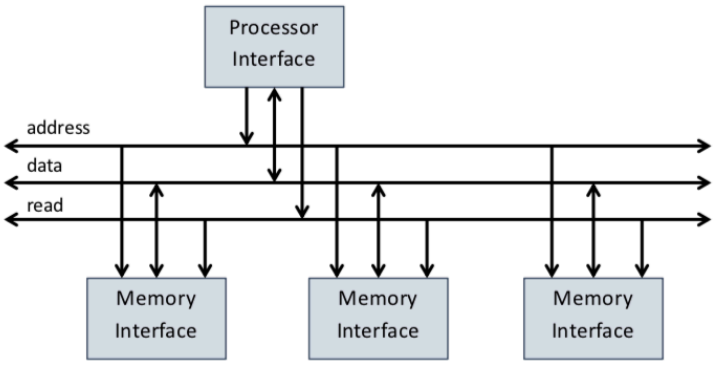
3. Create a SystemVerilog **interface** to encapsulate the interface signals. The interface should have clock and reset ports and provide two **modports**: one for use by the processor interface thread and the other for use by the memory interface thread. Do not make the ReadMem and WriteMem tasks part of the interface. Modify the top level module to instantiate the interface and modify the processor interface thread and memory interface thread modules to use the interface. Verify that it works.

4. Once the interface is working, modify the SimpleBus system to allow multiple memory interface threads (as loosely depicted below). As part of the change, use 24-bit addresses instead of 16-bit addresses. This will necessitate changes to both the processor interface thread and memory interface thread FSMs. Start by drawing new state transition diagrams for each. Strive to minimize changes to the FSMs (and SystemVerilog code).

   Each memory interface has a 16-bit (64KB) address space. The most significant 8-bits of the address indicate which memory interface will respond. You can think of the most significant 8-bits as the base address for a memory interface module's address space. It should be a parameter provided to each memory interface module instance. If an instance of a memory interface module detects its base address then it responds to the processor interface requests.

   Use a **generate** block to instantiate several memory interfaces, giving each of them a unique base address, beginning at 0 and incrementing by 1. The number of memory interfaces to instantiate should be a top-level parameter that can be overridden at elaboration/simulation time.

   Modify the rest of the design and verification code as needed to verify your changes.



Submit a zip file containing a directory (HW6) with the following files:

multiplypipelined.sv	Your pipelined multiplier using carry save adders (CSA)
fulladdr.sv	Your full adder design
multiplytb.sv	Testbench for your pipelined multiplier
decimal.txt	Some decimal testcases to demonstrate your testbench
hexadecimal.txt	Some hexadecimal testcases to demonstrate your testbench
edge.sv	A button synchronizer with edge detector
switchtail.sv	A 5-bit switch-tail ring counter
timeout.sv	A timeout counter
fsm.sv	A Moore style finite state machine to control the garage door
garagedoor.sv	The garage door system (FSM and other modules above)
fsm.pdf	Revised state transition diagrams for SimpleBus
simplebusif.sv	All the code for problems 3 and 4