David Greenberg

Project Checklist

1. CAM Buffer
    a) Working code for a CAM buffer
    b) Comprehensive test bench that tests adding and removing tags, modifying memory locations, and reading out all memory locations
        - Test bench will display messages when it begins and ends successfully, and it will print out an error message and terminate early if any test fails. The test bench will be written with tasks to make it clear what is happening at a high level to make it possible to understand what was tested.
2. Deterministic commit
    a) Test bench that checks that the round robin deterministic commit behaves according to the specification, where the earliest buffer in the round robin sequence that has written to an address is the one whose value is stored at that address
    b) Commit should be optimized for speed as much as possible to avoid stalls while scanning the buffers
        - Optimizations and the rationale for implementing them or leaving them out will be explained, along with documentation of any performance gains achieved
3. Arbitrator
    a) Arbitrator can forward read/write requests to the CAM buffer and stall the processor while the value is read from the shared memory location
    b) Test bench demonstrating both cases working
        - Test bench will display messages when it starts and ends, and it will terminate early and print an error message if any test fails. The test bench will be written with tasks to make

it clear what is happening at a high level to make it possible to understand what was tested.

4. Code style
    a) The code will be parameterized so that the number of CPUs and the size of the buffers are easily changed
    b) Repeated interconnect patterns will be generated with genvars
5. Modified micro8
    a) Supports being halted by the arbitrator
    b) This will be tested in one of two ways
        - A test bench runs the micro8 and halts it at various times and ensure that the computation finishes when expected with the correct results
        - The micro8 runs a simple program that displays graphical output on the screen and it is halted when a button is pressed down
6. Working system
    a) Demonstrating a system that is running one of the test programs (see below) in parallel and includes at least 3 CPUs and graphical output
7. Demonstration of determinism
    a) Test applications will be a race of threads writing to shared data and (possibly) a nondeterminism-sensitive pseudorandom number generator
        - Examples
            - see http://pages.cs.wisc.edu/~markhill/racey.html
            - There are several blocks on the screen that move up and down as the processors read and write to their corresponding memory locations. Every processor is writing to every memory location, and so the blocks will move in the same pattern after reset

only if the processors modified the memory deterministically

b) Test applications will display their output on the VGA port

c) Buttons on the lab kit will be connected to the processors, causing them to stall when the buttons are pushed. Regardless of which buttons are pushed or how long they're pushed for, the output of the application should remain identical