Hans C Anderson

hcander2

My testing strategy was to compile often and test as much functionality progressively as I was able. After completing cache reads, I used “basic.asm” to test and debug cache loads. In order to speed up time to generate waves, I created a .do file with the settings and signals that I found most relevant for debugging. If there were any timing errors (where there were a few), I would identify where the issue took place based on the wave form, then map out critical paths to make sure signals were in the correct places at the correct time.

Since there is lacking documentation on the syntax for LC3b assembly, I wasn’t sure how to test cache loads for eviction and LRU values until the test code was released. After the mp2.2 cp2 test code was released, I was able to ensure the correct lines were being evicted by walking through the code while viewing the wave diagram.

For cache writes, I was able to use “ldsti.asm” to double check that the basic load/store instructions performed correctly. Then I created a variation of the mp2.2 code called “dirty.asm” to debug write-backs. Bugs again required me to re-evaluate design decisions, rework ideas, and monitor critical paths to ensure correct operation. My largest asset to debugging was time.

By having the cache working before the test code came out, I believed that the processor and cache would perform correctly when I tested the program. I stepped through the test program instruction-by-instruction to ensure that all signals looked as expected. In this way, I found a bug in cache-write-misses due to signal timings. My last stage of testing was to walk through the code with the LC3 simulator while viewing the wave diagram to ensure that the registers received the correct values for the length of the program. This was already checked during the careful walkthrough, but helped to ensure there were no large errors that I missed.