OoO CPU: Testing Framework

December 19, 2025

1 Goal

The goal of this addition was to (1) remove the necessity to broadcast commit data on the CDB, and (2) set up mechanism to diff test vs spike. These changes are necessary to lighten CDB stress, and enable more rigorous tests than what was previously possible using waveform viewing and sv asserts on program end state.

2 Design Constraints

2.1 (1)

We want to increase the availability of the CDB for intermediate results, thus we must not broadcast commit data on the CDB.

2.2 (2)

There cannot be increased ROB width nuder synthesis, and no additional buffers to track trace data. We must percolate PC and instruction bits to the commit unit, to be logged by the testbench.

3 Solutions

3.1 (1)

Instead of broadcasting commit data on the CDB, we instead allow the issue unit to query the reorder buffer with the ROB number of the issuing instruction's operands. If the results are not in the reg file, and not in the CDB, then we check the ROB results.

3.2 (2)

We make use of the VERILATOR macro to gate extra ROB length for simulation. We add fields conditionally to the ROB buffer data type to include the PC and bits of the instruction. Then in the top level we conditionally expose ports that our test bench can use to print out commit metadata to be diff tested. For the testbench itself, we load the ELF into memory in a standard manner, and each cycle check if there was a commit and if so print the metadata in spike format. There is still more microarchitecture to implement before the `riscv-tests` repo can successfully be ran, namely CSRs and zicsr.

4 Tradeoffs

4.1 (1)

This solution increases the input width to the issue unit by a 66 bits (already fairly large) and the output width of the ROB by 66 bits. It also implemented with wiring from the register status register to the ROB.

This is worth it however because previously commits took precedence over every other CDB packet, as they were only available for one cycle and all else are registered. Thus we have decreased our CDB usage by commits/cycle. This is a massive improvement as commits occur about every other cycle for low latency operation workflows (no mul/div).

4.2 (2)

No real tradeoff here other than increased complexity / decreased readability of sections of the source code.