To implement option A of the design, we had to implement the RISC-32 Instruction Set Archiecture with a 4 to 5 stage pipeline for better throughput.

Our initial design in Senior Design I, called RAPID, showed promise to be an effective core design. Here, you can see the high-level block diagram of core. The block diagram was developed to help assist the development of the RTL code because for a project of this scale, keeping track of all connections can become challenging.

Part of the HDL design process is the design Finite State machines, here is an example of the memory stage FSM. You can see in the diagram, the “Wait for Pipeline” state which is used to synchronize operations between stages in the CPU.

Since we had developed our core early, we were able to begin early testing which provided critical feedback.

Early simulations indicated that our microarchitecture had race conditions relating to the register file. We also found that we had a relatively low Instructions-Per-Clock rate which risked our ability to meet our power and performance requirement. Given these observations we had decided to make another revision of microfracture to address the shortcomings of the RAPID architecture.

The optimizations done in RAPID-X microarchitecture, were realized by doing the following 3 things,

1. adding a forwarding unit
2. simplifying the finite state machines
3. And, reducing pipeline depth from 5 to 4 stages.

We were able to perform initial synthesize using Vivado, at this stage there were still many bugs in the core design.

Simulating the new core showed that we can reduce our minimum clock speed requirement from 350 MHz to 115 MHz to achieve our 100 MIPs goal.

The following benchmarks showcase a 300% uplift in IPC performance. For us to run these benchmarks, we had to create 4 test programs that covered branching, arithmetic, and memory access. We compared the performance with our original core, a reference open-source model, and our re-designed core.

Running these programs, we were able to simulate the pipeline latency. We measured a reduction of 9-cycles per instruction going from 12 cycles to 3-cycles.

Finally, we were able to synthesize the core using Genus which is part of Cadence tool-suite.