**Final Project Report**

The overall project included the design of a Single Cycle Processor which was then enhanced into a 5-stage pipelined processor with full forwarding and RF Bypass. In the third phase of the project, we implemented an I-Cache/D-Cache structure for the pipelined CPU along with a unified L2-Cache as demonstrated here:

A diagram of a computer

Description automatically generated

The implementation of the pipeline stages in the CPU is fairly standardized, however, several design choices were made for the caches that optimized for faster access and lower latencies. This includes a stack buffer table in the Cache Controller FSM that could keep track of more than one queued load/store requests (that were misses) along with the required pipelined word-flow necessitated by the project requirements. There was an attempt to completely decouple the load/store pathways in the CPU with other instruction pathways in order to facilitate all instructions independent of loads to pass through the CPU while loads are stalled for Cache Misses, but this could not be completed because of time constraints. This is perhaps why the D-Cache optimization is incomplete, although optimizations for I-Cache loading is complete including pushing an “LLB R0 0” instruction as a NOP during stall cycles due to cache misses.

We would create a table to describe the splitting of work in this project, but we have always dynamically allocated tasks based on requirements including debugging, thorough clear box testing, etc. and the tasks were split between us roughly fifty-fifty. However, to get into a little bit more detail, here is a more accurate split of the work:

1. Anna: Phase 1 she was responsible for designing almost the entirety of the fully functional ALU as well as the Control Unit of the Processor. For Phase 2 she was responsible for the development of the modified control unit as well as a few of the pipeline registers. For Phase 3, she developed the cache module, although work was split very evenly and dynamically for this part focused on completion as a goal.
2. Saro: Phase 1 he was responsible for designing all of the registers as well as bringing the whole CPU together to ensure proper functionality. For Phase 2 he designed some of the pipeline registers as well as the forwarding/RF bypass units. For Phase 3 he developed the cache control protocol as well as the cache wrapper and split trying to optimize the CPU with Anna, subject to severe time constraints.

Perhaps the most notable and impressive feature in the cache control protocol that wasn’t fully realized was the decoupling of the data path in the CPU to optimize non-loading operations that were independent almost mimicking Out of Order processing in a very limited superscalar structure. However, integrating this feature with the CPU seemed to be an extremely challenging task, which is why it was left incomplete. That being said, the Cache end of this implementation was complete and you can see a short stack to store such data in the controller module for superscalar structure.

For Phase 3 we regret to inform that we were unable to thoroughly test the test cases primarily because of the difficulty in integration of the cache wrapper module with the CPU. A lot of the features required significant tweaking and with a limited amount of time this was unfortunately not made possible. That being said, it is our request that you look into the waveforms of our processor to ensure that the caches do in fact work perfectly and given some time, the CPU issues would be fixed pretty seamlessly. Sorry about that.

For testing we typically would run exhaustive unit tests on individual modules followed by full-chip testing at the end of integration to figure out logic flaws in integration. For forwarding and stalling we exhaustively tested for almost all corner cases, including consecutive load instructions, for which WB-Mem Forwarding path was excluded. Further, an important corner case and design flaw in this processor was the NOP instruction, as the 0X0000 instruction tends to change the flag and hence flow of information in this processor. This design flaw was rectified in Phase 3 where new NOPs now consist of the instruction 0xA000, which is “LLB R0 0” and is known to have no effects on any part of the computer. This can be demonstrated pretty clearly in the following waveform diagram:

A screen shot of a computer

Description automatically generated

Notice how the IF\_ID\_instr\_out signal consistently reads A000 during the loading process. This indicates that a stall is inserted into this register continuously while the cache controller pipelines the loading process (you can see this between L2\_addr and data\_in signals). Right after this, not included in the picture is the actual execution of these fetched instructions, although truth be told, a design flaw in our D-Cache was the inability to “turn it off”. This resulted in the D-Cache thinking that it had a read signal even though there were no memory signals, and while it didn’t cause any immediate issues, it could definitely use some more work.

If any tests do not pass, this is primarily because of the integration, which forms a small part of the actual work behind this project but unfortunately might reflect negatively in the results. It is our hope that this is taken into consideration during grading.