MP4 Checkpoint 3 Progress Report: The Doug Dimmadomes

For this checkpoint, we ironed out some remaining bugs in our CPU implementation, and developed and introduced our advanced design features. Much of the debugging efforts were taken on by Michal and Max, and they worked to ensure the design-- without advanced features-- could properly execute the mp4-cp3 code and previous checkpoint codes without regressions. Many of the bugs were uncovered by peer code-review in which someone other than the original programmer looks over the code and suggests changes or clarifications to the code. This forced all of us to better consider design decisions and understand parts of the design one originally did not author. Michal discovered a significant bug in the arbiter unit which caused an incorrect memory address to be outputted when an instruction cache request and data cache request were issued concurrently, and Max discovered more minor bugs in the datapath and forwarding units which could introduced “don’t cares” into the pipeline and produced unintended behaviors. For the advanced features implemented for this checkpoint, Michal adapted the MP3 cache to serve as a foundation for a pipelined cache implementation. This advanced feature would increase the overall throughput of the design by allowing the memory subsystem to serve some cache requests immediately while others were waiting for data from physical memory. The testing strategy used to verify the pipeline caches was to develop small programs in assembler that would be easy to trace and compare the final register values against the same program executed on an MP2 or MP3 design. Derek implemented a pLRU and EWB for this checkpoint. The pLRU would improve performance by decreasing the amount of bookkeeping and hardware complexity involved in tracking usage of cache entries. Given a sequence of items and events acting over those items, the pLRU chooses to evict the item that is most likely to no longer have relevance. The EWB would improve performance by functioning as a temporary storage for dirty evicts from the L1 cache. Combined with the pipelined cache design feature, the EWB enables the CPU to serve cache misses right after a dirty evict and without waiting for a write-back to complete. The pLRU and EWB were tested using basic assembly programs and comparing the final register values with a working MP3. Max implemented a Return Address Stack and Tournament Branch Predictor for this checkpoint. The RAS would improve performance by removing the need to lookup the next PC value following a function call after a function returns. This feature was implemented by creating a multidimensional register and adjusting an index value pointing to the top of the stack when an op_jalr executes following an op_jal. Coupled with the tournament branch predictor, which improves performance by using information about previous branches using both a global branch predictor and local branch predictor to make predictions about whether the next branch is taken, these features improved the overall performance by minimizing the amount of times the pipeline needed to be flushed when a misprediction occurs. These features were verified using simple programs and comparisons with an MP3 design.

MP4 Checkpoint 3 Roadmap: The Doug Dimmadomes

For the next checkpoint, we intend to further optimize the design for the competition and iron out remaining bugs in the branch predictor and pipelined cache features introduced in this checkpoint. Also, we want to review the performance analytics for the design made available through Quartus to reduce critical paths and area complexity of the design wherever possible. Michal will lead the debugging efforts and remaining advanced feature design integrations, and Max and Derek will focus on simplifying various

aspects of the design according to the analysis tools and optimizing performance for the competition codes.