William Ma (Partner: Nicholas Petosa)

Professor Hyesoon Kim

CS 3220: Processor Design

17 April 2017

In terms of workload, again Nick and I each did 50% of the workload, with my work primarily being focused on the performance optimizations of the project, including data forwarding, reducing critical length/times, etc, trying to improve IPC, and finally the overclocking of the PLL that resulted in the more noticeable performance improvements. Nick on the other hand focused primarily on implementing I/O devices and that section, with each of us writing about 50% of xmax.mif. Performance optimizations were tricky from the get-go, with data forwarding being annoying for some time because of oversights and implicit relations in the frame we were provided, which, once made explicit, were resolved very quickly, bringing us from a 60 second execution time to 47 seconds. With the PLL overclocking that I performed after all that, we were able to reduce the execution time of fmedian2.mif from that 47 seconds to a cool 26 seconds, meaning a 57% reduction in execution time overall, a very noticeable improvement. We did not implement a sophisticated branch predictor to try to improve this time, as we noted through dynamic instruction counting that about 99% of the branch statements in fmedian2.mif were not taken. As a result we kept our static not taken branch predictor, as we could not well improve upon that 99% accuracy. Note that we managed to overclock the PLL to 90MHz from a reference clock of 50MHz, with no negative slack concerns or timing issues.

There were a number of issues that came up was we were working on this project, the largest time-wise coming up during the implementation of data forwarding, largely a result of the design choices we made in the previous project, i.e. using an isnop flag bit instead of just clearing registers, which produced forwarding problems, as we were getting false dependencies. The way to solve this was a hybrid clearing and isnop approach, which floored registers to zero for nops and, since the zero register is never written to and should never be forwarded as a result, meant that those nops were finally properly ignored by the data forwarding controller that we implemented.

The next big chunk of time came while implementing devices, which took a substantial amount of time, since we had to develop it in parallel with our xmax.mif. We weren't sure in the beginning whether we were going to implement interrupts or not, but largely because of time constraints from other deliverables we decided to go with a hardware polling approach, which is reflected in the xmax.mif, from the fact that we are polling a hardware ready bit in order to know when to change the LEDR display. Similar logic follows for how the HEX display is updated in xmax. In the end, we have a functioning pipeline that takes almost one third as much time as our project 3 pipeline did, with much more functional devices support. That should suffice.