

(a) What did you learn from this project?

- This course gave us a conceptual understanding of the basics of micro-processor hardware design. The project attached to this course gave us an opportunity to go one step further and design our own processor and thus taught us to implement the concepts in designing the processor hardware. We learnt that out of all available data-path designs, the pipelined approach is the best in terms of processor performance. Although it may involve a complex data-path design, once implemented will give the best results.
- We started off with structuring a single-cycle data-path and then advanced it to a pipelined one by adding the pipeline registers. We ensured that the entire instruction set was executable on the designed data-path. We had to re-visit the designed data-path to make a few changes as per the errors encountered in the verification stage. Overall, the pipelined data-path is not as difficult to implement as it seemed in the beginning.
- We also learnt that working in groups is more efficient as there are two sets of eyes looking at the same problem. This enhanced the design and code written for components. De-bugging is the most important stage as this will ensure less head-aches in later stages.

(b) What would you do differently next time?

There are several things that we would do differently next time:

- Design instruction set for specific purpose applications, while maintaining support for higher level constructs.
- Implement Hazard detection and correction: After learning in class, we learnt that hazard detection and correction are a must in every pipeline data-path as it saves cycle penalties and avoid insertion of multiple no-operation cycles. We would definitely start off early in implementing these units in the pipelined data path.

(c) What is your advice to someone who is going to work on a similar project?

- Start earlier: The coding for components and verification of the entire data-path are time consuming processes, taking approximately 2 weeks of precious time. This will indirectly help in staying ahead in course work. We waited for some concepts to be taught in class before implementing them. Try and learn these ahead of class.
- Coding: Improve familiarity with VHDL coding language, methods and syntax early. This will help in saving a lot of time debugging unwanted errors in program structure, syntax etc.
- Approach the GTA (Manish Kulkarni, if he is still your GTA) early for the FPGA implementation part. He is very friendly, helpful and is thorough with the tools.