Here are some guidelines for projects whose goal is to write architecture-conscious code for WaveScalar. We can develop the same sort of guidelines for other projects.

1. Choose an application. Either use one we have found (they will be on the course web pages) or one of your own choice. The latter might be an algorithm from your probably research area (if it is not computer architecture, that is). We already have ideas from computational biology, vision, and graphics. Why do you think your choice is a good program to execute on a spatial dataflow machine, such as WaveScalar? Let Andrew know what you have chosen, as an early sanity check.
2. Profile the application to determine its hotspot code. What percentage of the time is the hotspot executed? Is it small enough to code in WaveScalar assembly if necessary? Would the second hottest spot be better (not as hot, but smaller)?
3. From eyeballing the source and understanding what the algorithm does, can you tell if there are independent operations within the same iteration? Across iterations? Are there independent memory operations within the same iteration? Across iterations? How (in terms of operations) big are the iterations? Are the same data memory locations used by multiple operations?
4. Milestone: discuss your application & the reasons for choosing it with me or Andrew.

5. Compile the hotspot code using WaveScalar's binary translator. Is it necessary to compile the whole program, or can you construct a wrapper for the hotspot code and just compile that (like was done in `lcs`)? Try to do the latter.
6. Simulate the hotspot on the WaveScalar simulator. How fast does it run? How much parallelism does it exhibit? What is the performance of the memory system? Are there any other metrics you think it important to gather before beginning the WaveScalar-friendly design? Do they need to be added to the WaveScalar simulator?
7. Milestone: discuss the performance of what will be the baseline for your algorithmic designs with Andrew and me (make an appointment or come to office hours).

8. Rework the hotspot code to make it WaveScalar-friendly. Some avenues you might explore are:
   - Exploiting more parallelism, i.e., creating more threads and, in particular, lightweight threads.
   - Using fewer memory operations, i.e., point-to-point operand transmission rather than reloading a value.
   - Using unordered memory operations for code that has no data dependences.
9. How much faster is the new code and why?

Now there are alternative paths you could take. Things of this nature may or may not be doable in a quarter.

10. Are there other ways in which you could improve the performance of your code? What is your evidence for thinking this? (Make sure you have experimental evidence before embarking down this path. But if you do, what about:

- Changing the configuration of some key components of the WaveScalar microarchitecture (your hypothesis spells out why your application is sensitive to the design of this component, right?)
- Adding new components to the WaveScalar microarchitecture (same)
- How well does your new WaveScalar microarchitecture execute other programs? We have applications from SPEC, SPLASH2 and Mediabench you can use.

11. Put the hotspot results in context by comparing them to those you get when the hotspot is simulated in its application.
12. Most parallel machines today can only support coarse-grain parallelism (or ILP), because inter-thread communication is so expensive. WaveScalar is currently the only architecture which can support granularities of parallelism of arbitrary size. Take advantage of this by coarse-grain parallelizing the application and fine-grain parallelizing the hot spot within each coarse-grain thread.
13. WaveScalar may or may not benefit from traditional compiler optimizations, which usually assume a target machine that looks like a superscalar. For example, it may be better to repeat calculations than apply common subexpression elimination and ship the resulting value to all consumer instructions of the expression. Implement the effect of a compiler optimization on your code. Does it help or hurt, and why?

14. Milestone: discuss your new design & results with Andrew and me (make an appointment or come to office hours).

15. Write the report.