CSE471: Computer Design & Organization
The Project
Due: check project milestones in the calendar

The purpose of this assignment is give you experience in investigating and carrying out more open-ended topics in computer architecture, very similar to what you will do in research or advanced development both in academia or industry.

For the rest of the quarter and in small groups of two or three, you'll work a single project. Below are suggested topics for this project, but you are welcome to come up with one of your own:

1. Hardware accelerators for improving the performance of critical code.
   The goal of this project is to evaluate the costs and benefits of special purpose, application-specific hardware. Choose an important application which is open-source, so that you have access to the code. Profile the code (gprof) to determine the hot functions. Through a combination of finer-level profiling (Intel's Pin: http://www.pintool.org/) and examining the source, try to figure out where hardware support might speed up the application. Design that hardware at the microarchitecture level. Justify the potential hardware cost and performance benefit of your design. Is there a way you can implement your design in just the short time we have to provide a quantitative backing to your justification?
2. Avoiding unnecessary synchronization.
   The goal of this project is to investigate whether there are applications in which synchronization is not really necessary. Find an open-source parallel application where data dependences might not be crucial to acceptable execution. Good examples might be video compression/decompression in media applications or ray tracing in graphics, but try to think of others. Determine the performance of these programs on one of the department's parallel computers. Use the performance hardware to determine component values of that performance or any other measurements you think useful. Use Intel VTune to evaluate the cost of the synchronization in some way. Devise a strategy to remove the synchronization from the programs and reevaluate. What is your conclusion regarding the trade-off between performance and quality of the results?
3. Designing and evaluating cache coherency protocols.
   The goal of this project is to design and evaluate a cache coherency protocol that is better than the three-state, invalidation-based protocol we'll study in class. What would be a useful fourth state? Profile a few parallel programs (using Pin) to get a handle on whether your fourth state is useful. Implement both the three- and fourth-state protocols and evaluate. Was your fourth state a good idea or not. Why or why not?
4. Microarchitecture support for Semantic Memory.
   The goal of this project is to design a mechanism for memory-access-triggered computation for Semantic Memory. For background on Semantic Memory read the

Semantic Memory grant proposal and for a possible starting point for your design read the research paper on Informing Memory Loads. Would you use a generalization of the mechanism in Informing Memory Loads for Semantic Memory? Is another approach better? How would you make accessing both the data and a pointer to code that should be executed (triggered) fast? How would you handle the common case in which there is no pointer? What is a good object granularity for the triggers: pages? individual data structures? How does that affect your implementation?

5. Semantic Memory Tags.
   The goal of this project is to understand and critique past research on memory tagging and to design tags for Semantic Memory. As background read and critique the following papers on tags (Mondrian Memory, Raksha, MemTracker, Loki). If Semantic Memory tags were to be located in memory and you wanted to use them for cache coherency or data consistency, what would the tags contain and what would be their design?
6. FPGA-based Simulators.
   Often we need to obtain performance results over execution runs that are just too long to do in software simulators, such as tracking rarely occurring events in an operating system or executing an entire application. The common thinking among experimental computer scientists, at least architects, is that running the experiments on an FPGA is a better methodology. But just how easy is it to roll out an FPGA-based design of a processor? Test the hypothesis by trying to "boot" and then alter the Sun open-source synthesizable SPARC processor core (called OpenSparc) on an FPGA board. What was easy as pie? What was difficult? What do you think about the current wisdom on FPGA-based simulators? Run experiments to evaluate performance on the FPGA relative to instruction-level simulators which usually run at a few tens of thousands of instructions per second (depending on the level of detail). You might want to add some sort of performance monitor to enable you to evaluate component metrics of the bottom line. Someone on this team needs to know HDL programming.
7. Programmable ALUs.
   This project aims to determine whether, if you had the fanciest ALU you could design, i.e., if you had an ALU that was implemented as an FPGA, what would be the resulting performance. The ALU should be able to combine any logical and arithmetic operations into a single-cycle function. By examining the assembly language version of an application, can you identify short sequences of code that operate on the same input operands? To save your eyeballs, can PIN be used to detect appropriate instruction sequences? Can you devise a way to evaluate your super ALU? If so, go for it! Background reading for this project would be the PRISC and VEAL papers.
8. Data regions for Many-Cache FPGAs.
   The many-cache memory model was developed to better utilize the distributed BRAMs on an FPGA by organizing them into caches that are local to their computation and customized for the memory access patterns in an application. The compiler algorithm that constructs many-cache caches needs to divide main memory into separate, independent regions (e.g., two arrays that don't overlap can go into two separate memory regions). Choose a set of applications that have a hot

kernel (such as those in the Splash-2 or Mediabench benchmark suites) and try to decompose and reconstruct the code and/or data to create separate memory regions. Are the clever ways to divide arrays and data structures so that they can be placed in separate memory regions? Can you devise a methodology to evaluate the reworked applications and compare them to the original? Do we have the tools in hand to do that evaluation? If so, be my guest. Background reading would be the two CHiMPS papers.

First decide how you are going to go about carrying out your project. What questions will help you investigate the issues? How will you go about answering the questions? What papers will you need to read to obtain good background in the area? What software or hardware infrastructure will you need? Which parts of this infrastructure will you need to develop yourselves? Vince and I will help you frame this in the initial project group meetings on Thursday, April 21.

There are several milestones along the way to keep you from falling behind. In the first milestone you'll define your project. This is a two-pager which will basically answer the questions above, i.e., will describe what you'd like to do and how you're going to go about doing it, complete with a timeline of what will be accomplished at each milestone. For the next two milestones you'll be doing the bulk of the work. And the last milestone is for writing up the final report on your project.