I am applying to graduate school to explore how best to physically implement computation and how best to pass on what I learn to others. As physical devices get pushed to their limits, new design paradigms have begun to emerge across the spectrum of hardware architectures. This spectrum spans from prominent software-programmable devices, such as many-core CPUs and modern GPUs, to hardware-programmable technologies such as FPGAs. I am fascinated with the design challenges inherent in these and other types of devices and want to contribute to meeting these challenges. My eventual goal is to share what I learn with others, as a professor. Discovery and education are equally important at a time when computer science as a field is advancing a great deal. I want to contribute to this ongoing revolution by learning about different ways to physically implement computation, while giving back to the current and next generation as an educator.

How to implement computation involves design decisions from the application to the hardware architecture level. As non-parallel CPU performance is no longer scaling with time, these devices are evolving to support higher core counts and more elaborate system interconnects. I want to learn how these next-generation compute engines can be designed to support next-generation applications. Furthermore, I would like to study how novel architectures can support different applications sharing similar computational needs. Novel architectures can theoretically build on any idea taken from general-purpose compute engines to fully custom accelerators. I believe that the optimal solution will find common ground between the two. Moreover, I believe that with parallelism becoming ubiquitous, applications will again drive design at the hardware architecture level. To match physical computation to applications, one must consider the entire design space between traditional generalized architectures and fully custom accelerators.

Attempting to occupy the space between general-purpose architectures and custom accelerators brings to mind several questions. The first is how to architect devices to carry out computation that best fits the needs of various families of applications. The second is to what degree, when a device cannot meet the demands of some of its applications, other means of computation can be introduced to better meet design criteria. I want to consider the entire hardware architecture spectrum to answer this family of questions: how best to implement computation when it can change dramatically in form, given different applications.

During my undergraduate career, research experience has given me the background to explore these problems in more depth as a graduate student. This year, I have been working with Professors John Wawrzynek (U.C. Berkeley) and Garry Nolan (Stanford University) to create an FPGA-based accelerator for learning cell signaling network structure. The overall purpose of the approach is to provide a human-interpretable map of how cancer cells process information, and in so doing grow beyond their healthy confines or resist drug treatment. According to Dr. Nolan and his colleagues that I have interacted with, this will allow researchers to create novel diagnostics or guide therapeutic advances for patients. Mapping this problem to hardware is desirable because the learning algorithm's complexity is super-exponential with the number of interrelated proteins in a cell. One of my contributions to the project was to optimize a computationally expensive part of the algorithm, built on an FPGA. Given that we were designing to a specific algorithm, our design could achieve speedup by taking advantage of both micro-architectural optimizations, such as parallelism and multi-threading, and characteristics of the FPGA fabric itself. By designing with both the application and device in mind, we were able to

develop a highly optimized solution to an important problem in systems biology.

Working on the signaling networks project has had a great impact on me as a student and a researcher. The project has given me a broader perspective on my work as a computer scientist and how it is relevant beyond computer science as a field. I have seen how it is desirable to take a theoretical advance in multiple fields and apply what was learned to a real world problem, that would otherwise (according to Dr. Nolan and his team) be stuck in low gear because of computational limitations. Specifically, fields involved include systems biology, statistics, and computer science. I recently gave a talk on the project[1] at the Gigascale Systems Research Center (GSRC). Giving the talk, and seeing how the audience responded to it, inspired me. Working on this project has shown me that research is about answering questions that we care about as a civilization.

Given the hardware architecture spectrum, and different problems which motivate its growth, a question of great personal importance is how best to communicate these concepts to others. Consistent with a research movement led by the U.C. Berkeley ParLab, one approach is to describe complex systems in terms of design patterns.[2] A design pattern describes a certain type of problem or family of computation, and points to known techniques used to implement that computation. I have had the privilege to teach lower schoolers, middle schoolers, high schoolers, and college students across a range of subjects in the past and all have been able to learn from, and relate to, patterns. I have found this to be especially true in teaching students about computer science, as design patterns allow complex systems to be decomposed by algorithm and execution model. As a teaching assistant for an introduction to digital design class, for example, I was able to discuss complex micro-architectures by reducing them to concepts, or patterns, relevant to digital design. In my opinion, discovering how best to explain computation is just as important as learning how to implement it.

Exploring the spectrum of hardware architectures requires expertise and dedication from many disciplines in computer science. Berkeley's demonstrated commitment to both applications and hardware architecture, as well as its exemplary faculty make it the ideal environment for me to conduct my graduate study. In the realm of computer architecture, I feel that I would be able to learn a great deal if given the opportunity to work with Professor XX. Professor XX's architectural vision, as was proposed this semester, along with his take on hardware design in the form of patterns, is very important amidst the transition to the many-core architectures that I have discussed. I would also be very interested in working with Professor YY. I've known Professor YY from the perspective of a student, teacher, and researcher. Just recently, he announced a fast-paced vision which (I have to say) was very exciting to be a part of. Professor YY interests in a variety of platforms, and the paths that can be drawn between them, illustrate a new way for me to think about the material that I want to work on, looking forward. While I have mentioned several faculty whose work is particularly interesting, I look forward to getting a closer look at several other research teams in which to potentially pursue Ph.D. research. I realize that at the end of the day, I have to choose one or two project areas. Regardless, I hope that these areas can be in an interdisciplinary environment that helps drive innovation.

[1] Talk slides can be found at http://cwfletcher.net/GSRC2009.pdf.

[2] The lead projects are OPL ("Our Pattern Language") and BHPL ("Berkeley Hardware Pattern Language").

In summary, my goals as a graduate student are to research novel hardware architectures in order to solve next-generation problems and to learn how best to communicate these challenges and proposed solutions to others. Looking back as a teacher, one challenge continues to be emphasizing the connection between the application and device. There is pedagogical value in decoupling the two, as doing so favors development simplicity. I believe, however, that even high level developers must always keep their underlying platform in mind. As computation evolves to match device capabilities and different applications' needs, only in respecting both domains will we be able to continue the trend of high-performance over time. Looking forward, I am excited to have the opportunity to contribute to addressing this challenge.