

I have been fascinated with the future of computer engineering when considering its increasing reliance on integrating skills in physics, devices, computer architecture, compiler optimizations and theory. This merging of technology fields is not a new concept, but it seems to be growing in importance. As a result, it may be partially responsible for a shift in the job market for many computer science graduates as traditional programmers no longer seem to be a rare commodity. When confronted with the idea of teaching computer engineering courses, I realize that after completion of the core computer science curriculum, it is important to give students both time and resources to apply their computational skills to fields outside the boundaries of their particular major. For example, a specialized target course for computer architecture students, can be oriented at creating architectural extensions to allow easier virtualization software integration. Even in a multidisciplinary approach to teaching, however, it is important to keep the students' interests focused. In a scientific environment that changes almost daily, unless students focus on specific threads of specialization important to them, it can be very hard to achieve a level of expertise required in any particular field.

As a graduate researcher, I have been very fortunate to join a lab that embraces the multidisciplinary approach to both teaching and research. Our group has produced papers at some of the most respected computer architecture venues on topics that range from quantum computation to analyzing the behavior of internet worms, where each paper is led by researchers with a strong interest in the corresponding topic. In all projects, creating the necessary design tools had required the close collaboration with specialized research groups all across the nation. As a research advisor, I hope to foster similar collaborations that will allow my graduate students to obtain experiences that are invaluable when entering diverse scientific markets after graduation.

I also hope to bring similar philosophy to my teaching repertoire. As much as computers are the same in their core design and functionality, new ideas and configurations are experimented with nearly daily. I can see the concept of teaching novel computing architectures to both graduate or undergraduate students as being divided into several smaller "sub-courses". Each "sub-course" can be designed to give interested students direct experience with few of the most cutting edge design issues in modern computer architectures. Without commitment to the actual course titles, there are three "sub-course" categories that I am interested in teaching to graduate or undergraduate students in addition to the required core curriculum. The three categories are:

Emerging Computing Architectures: The intent of this course is to provide the students with a window into the system-level issues that are present when considering new and emerging computing device technologies from science fields such as quantum physics, optics, biology, and chemistry. The students will become broadly familiar with alternative concepts for storing, processing, and transporting information and will understand the prevailing system-level issues with new devices, such as fault-tolerance, scalability limits, and the limits of the application space. Knowledge about the information processing and storage capabilities of molecular based electronics, self-assembly nano-structures, and optical interconnects will provide the students with additional intuition about technologies who will not necessarily compete with CMOS, but rather offer the ability to process application domains not accessible to CMOS.

Fault-Tolerant Hardware Mechanisms: The increasing scaling of transistor devices has led to significant amount of research in hardware mechanisms for detecting manufacturing and run-time device errors. I believe that the design of rigorous fault-tolerant architectures is crucial not just for emerging technologies, but for scaled CMOS devices whose error rates due to instable manufacturing costs and increased wear-out may undoubtedly reach those of proposed nanoscale devices. The course will be a thorough literature survey where students can hopefully offer surprising and interesting new ways to expand on the existing fault-tolerance themes.

Large-Scale Quantum Computing Systems: Last, but certainly not least, the design and modeling of balanced quantum architectures, that utilize concepts of traditional computers is a course I can see myself teaching. This course is important on two levels: 1) It leads the students to apply their knowledge for traditional architecture design, such as specialization, to developing reliable large-scale quantum architectures whose application domain is inaccessible to current computers; and 2) The course allows the students to personally develop tools for modeling quantum architectures and perhaps come up with a novel way to control fault-tolerant systems through compilation methods that will apply to traditional CMOS devices. My first significant experience with this course will be applied at a tutorial in the 39th International Symposium on Microarchitecture (MICRO). The tutorial is based on my book titled "Quantum Computing for Computer Architects." which focuses on key architectural issues for designing balanced and feasible quantum computers.