Modernizing the Computer Architecture Curriculum at Carnegie Mellon: A Multi-Core-Systems Centered Approach

Onur Mutlu (onur@cmu.edu)
Carnegie Mellon University
http://www.ece.cmu.edu/~omutlu

The Setting: Computer science and engineering is undergoing a revolution. Computationally very powerful *parallel computers*, which used to be the luxury of the government and billion-dollar corporations, are already in the laptops and desktops of millions of ordinary users. Computer architects are building existing computer chips with *multiple processing cores* inside them, as opposed to with solely a *single processing core*, which used to be the traditional way of designing mainstream computer chips until around 2004. Essentially, a processing core is analogous to the brain of the computing system: the more cores there are, the more tasks the system can perform in parallel. Chips with multiple processing cores are commonly called *multi-core chips*. Existing Intel and AMD chips in the market already have 4 cores, IBM and Sun Microsystems have chips with respectively 9 and 16 cores, and Intel has demonstrated prototypes of an 80-core chip. Both academic and industrial researchers, including us at Carnegie Mellon, are envisioning and charting out designs of 1000-core chips in the 10-20-year timeframe [1, 2]. Soon, unprecedented amounts of computing power will be in the hands of almost every single computer user and programmer. And, both the programmers and the users need to be aware of how to harness this power.

To aid understanding, Figure 1 shows an example single-core system and an example multi-core system with nine processing cores. Major differences between the two systems are highlighted in terms of designers', programmers' and users' perspectives.

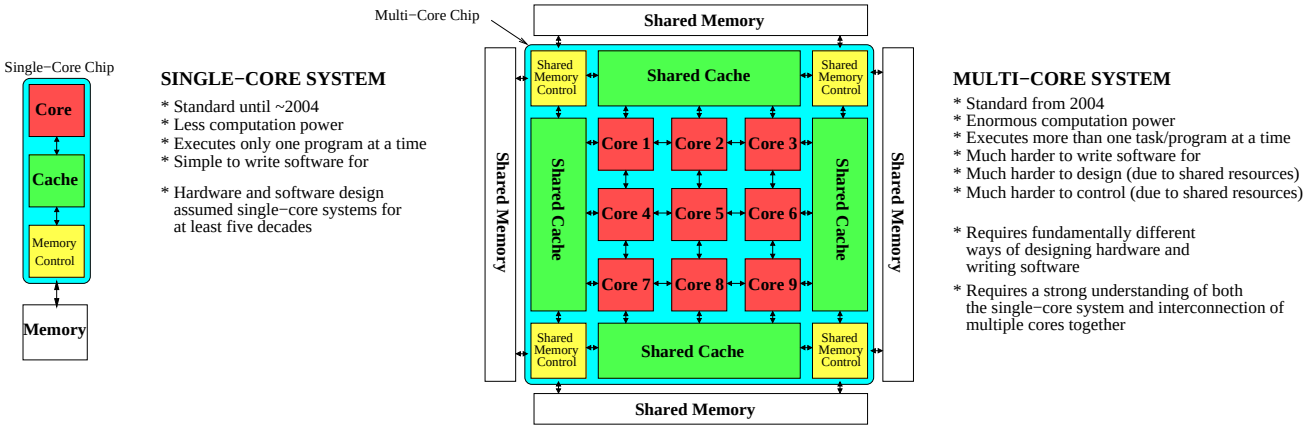

Figure 1. Illustration of the difference between traditional single-core and modern multi-core systems. The figure shows a nine-core multi-core system on the left; future systems are projected to have hundreds and thousands of processing cores.

The Problem: Computer architecture is the science and art of understanding, designing, and interconnecting hardware components and designing the hardware/software interface such that the resulting computer chip/system satisfies specific performance, power consumption, energy-efficiency, and reliability requirements. Traditionally, *computer architecture education and practice have largely assumed that there is only one single processing core on a chip*. The assumption stems from the fact that software programs that execute on the hardware were *sequential*, i.e. had single thread of execution. However, with multi-core chips, this assumption is violated. To obtain satisfactory and scalable performance from multi-core chips, not only should the chip/system be architected from scratch with keeping the parallel execution of multiple programs in mind, but also software should be re-designed from scratch to use multiple cores. This changes the way in which fundamental computer architecture concepts should be thought of and taught, as well as the way in which fundamental programming and software design concepts are thought of and taught. Literally every aspect of computer architecture design and thinking changes with the existence of multiple cores instead of a single one (Figure 1 provides a glimpse of such changes). *If we would like future generations to truly understand how current and future computers work and if we would like future hardware/software designers to push the boundaries of hardware/software design, we, as educators, need to make multi-core architectures a central part of the computer architecture and systems software curriculum.* But, is our computer architecture curriculum ready to prepare the students adequately to understand, design, and program multi-core systems?

Existing Computer Architecture Curriculum at Carnegie Mellon: Current major computer architecture courses (18-447, the undergraduate course, and 18-741, the advanced graduate course) in the ECE Department at Carnegie Mellon cover almost exclusively single-core architectures. Multiprocessors (in the traditional sense, where single-core processors are simply connected instead of placing multiple cores on chip) are very briefly covered in one lecture in the graduate course, but not in enough detail and emphasis to give students a strong understanding of existing multi-core systems. Even the parallel computer architecture course, 18-742, which was last taught in Spring 2006, focuses on traditional *multiple single-core chips* instead of *multi-core chips*, which are fundamentally different from each other [1, 4]. As such, existing core courses at Carnegie Mellon do not adequately prepare students for industrial or research jobs in which they will have to design the hardware or software for multi-core chips.

Our Goals: Carnegie Mellon has traditionally been a powerhouse in computer architecture education and research in the world. Our goal is to re-design the computer architecture curriculum such that we adapt our curriculum to the multi-core revolution and continue to keep our edge in the forefront of computer architecture education and research by providing the necessary background and skills for students to design and innovate multi-core systems. To achieve this goal, this proposal aims to accomplish the following synergistic sub-goals:

1. One of our major goals is to **re-design the existing fundamental computer architecture courses** such that their focus is on the design of multi-core systems rather than single-core systems. The focus will initially be on graduate-level courses (18-741 and 18-742), but eventually (not as part of this proposal) the undergraduate-level computer architecture course will be re-designed as well (18-447). The major theme of the re-design, which is elaborated below, is to start the course with the assumption that existing computers are multi-core computers and to teach all fundamental computer architecture concepts (even those related to single-core systems) within the context of a multi-core system. We believe this re-design of fundamental courses will adequately prepare our graduates to jobs in industry as well as basic research in computer architecture.

2. The second major goal is to **develop a focused graduate-level course on the hardware/software design of multi-core systems**. The purpose is twofolds. First, to teach the Ph.D., masters, and motivated undergraduate students the fundamental challenges and research problems in designing hardware and software for multi-core systems, and the basic solutions to them as we know now. Second, to challenge the students to develop out-of-the-box thinking and solutions to the described problems via a focused research project where the students will work in groups. We believe such a new course will prepare our students to do advanced research in multi-core systems related topics in hardware as well as software.

3. The final major goal is to **develop synergistic activities to foster broader multi-core education** to enable Carnegie Mellon to be a leader in "mainstream parallel computing." This consists of two components. First, we will work with other faculty members to incorporate concepts of multi-core systems into related courses. The software stack that runs on hardware is very much affected by the movement of hardware from single-core to multi-core and therefore needs to be re-designed and re-thought. Operating systems, compilers, user interfaces, and algorithms are all affected by the multi-core revolution. Incorporating multi-core concepts in such courses will enable our students to stay up-to-date with current technology. We will work with faculty members teaching these courses to devise a comprehensive plan for incorporating multi-core revolution and thinking into the courses in a consistent manner. Second, we will educate the broader Carnegie Mellon community by giving accessible seminars within the university about the potential and caveats of multi-core systems as well as inviting prominent speakers from industry to give similar seminars. The widespread availability of very low-cost and very powerful parallel computers can enable sophisticated uses of technology in other fields than engineering, such as fine arts, motion arts, natural sciences, business administration, and social sciences. Our broader education will be aimed at enabling professionals and students in these fields to make the best use of multi-core computers for their purposes without requiring them to know about the underlying technical details.

Teaching Philosophy (as related to this proposal): A cutting-edge school needs to educate its students and staff in cutting-edge concepts. There are two aspects to this education. First, the fundamental concepts need to be taught strongly so that students can acquire the necessary skills to think independently without being restricted to the state-of-the-art. Second, the state-of-the-art in a field needs to be woven tightly into the teaching of the fundamental concepts such that students acquire a strong perspective of the existing approaches to fundamental problems and develop the ability to improve cutting-edge concepts. These two aspects can be seen as conflicting, but they do not need to be. In my teaching, my goal is to hammer home the fundamentals while providing the students the necessary contemporary perspective of cutting-edge practice and research. As a concrete example, *caching* is a fundamental concept in computer architecture that has been proven to be useful over 50 years of research and practice. Almost all systems built since the early 1980s included the concept of caching by implementing hardware-based caches to improve system performance (as demonstrated pictorially in the single-core system depicted in Figure 1). Almost all computer architecture courses in the country teach caching within the context of a single-core system. However, the fundamental concept of caching requires a very different way of

thinking if we consider a multi-core system instead of a single-core system. In a multi-core system, multiple processors share the hardware-based cache, which fundamentally changes the way caching is designed, implemented, and evaluated to achieve high system performance. Figure 1 pictorially shows the differences between a shared cache in a multi-core system and a cache in a single core system: the cache in the multi-core system is distributed into multiple pieces, and each piece can be accessed by each core via a network of wires. In fact, even the definition of "optimal caching" changes when we move to a multi-core system from a single-core system [5]. If the students are taught the fundamental concept of caching from a single-core perspective, they will not be prepared for the reality of multi-core systems, which is the state-of-the-art in computing. For this very reason, my teaching philosophy is to combine the state-of-the-art practice and research tightly into the fundamental concepts such that students are prepared for making a difference in both real-world designs and cutting-edge research.

In the broader perspective, I see a teacher as an educator not only in the classroom but also in everyday life. The mission of an educator should be to lay out the concepts as clearly as possible such that non-experts in a field can make use of the provided information. Computer architecture and multi-core computing are areas that affect every member of our university community, directly or indirectly. Educating clearly our community members on what they can accomplish with multi-core computing at a level they can empathize with is an important teaching responsibility that I aim to undertake within the scope of this proposal.

Activities to Meet the Educational Goals: As outlined above, this proposal aims to achieve three major educational goals. Hereby we briefly describe the concrete steps we are taking and we will take to achieve each of the goals.

I. Re-design of the Advanced Computer Architecture course (18-741): The author is currently teaching 18-741 (in Spring 2009). During the course of teaching, each lecture is being designed from scratch such that it meets the following criteria:

1. The lecture briefly introduces the fundamental concept being taught (e.g., caching, as described above) as it relates to single-core architectures.

2. The lecture describes how the fundamental concept has changed within the context of multi-core architectures. Depending on the concept, this is where a significant portion of the lecture is spent.

3. The lecture describes open research and implementation challenges related to both single-core and multi-core aspects of the concept.

Structuring each lecture as described allows us to teach both the fundamental concepts as well as their multi-core aspects concisely. Homeworks, programming assignments, examinations in the course are also accordingly re-designed to break the assumption of single-core systems that used to be present in the course. 18-741 also includes a major design project in which students, in groups of two, perform a research project in a computer architecture topic. The project topics are being re-designed such that they are all multi-core oriented. In fact, thirteen out of the sixteen project groups are currently doing projects that are very tightly related to multi-core systems. Our hope is that focusing on multi-core systems in 18-741 will enable graduate students to 1) quickly jump into cutting-edge research in the field or 2) quickly get used to designing software and hardware for multi-core systems upon graduation, without going the extra step to learn about multi-core systems on their own (or via other means).

II. Re-design of the Multiprocessor Architecture course (18-742): This course was a course focused on designing multiprocessor architectures by connecting single-core chips together. With the arrival of multi-core chips, this way of interconnecting single-core systems to form a multiprocessor is no longer the mainstream way of building a parallel computer. Having multiple cores on the same chip enables many new optimizations and software/hardware opportunities (e.g. fast communication between cores, on-chip networks, fast scheduling of tasks) that is not possible in traditional multiprocessor systems. However, when 18-742 was taught last time, multi-core systems were covered in only two lectures toward the end of the semester. Our goal is to re-design 18-742 such that it assumes multiple cores are on the same chip. To accomplish this, we aim to re-design each lecture such that:

1. the concepts are first introduced within the context of a multi-core system with an on-chip interconnection network.

2. the advantages/disadvantages of a multi-core system is clearly described in comparison to a traditional single-core based multiprocessor system.

3. research and implementation challenges in building scalable hardware for multi-core systems are outlined.

The homeworks, programming assignments, and examinations will also be re-designed with a fundamental emphasis on multi-core systems. We would like to overhaul the research project in this course, which assumed extensions to traditional multiprocessors and simulation-based evaluation of such enhancements. The focus of the project will be shifted to 1) enhancing multi-core architectures in which students will propose changes to the state-of-the-art, 2) evaluating the enhancements using hardware-based prototyping of multi-core architectures, especially the hardware-based modeling of shared caches and on-chip multi-core interconnection networks. The former will enable the students to work on state-of-the-art and the latter will lead to more accurate evaluation results.

III. Development of a focused graduate-level course on hardware/software (co-)design of multi-core systems: Multi-core revolution has enabled the possibility of optimizing the computing system hardware and software at the same time because neither traditional single-core hardware nor traditional single-core software can work in a multi-core context. Therefore, a major opportunity exists to re-think the fundamentals of computing such that hardware and software are co-designed in a general purpose system to enable higher performance [6]. In fact, as many researchers have noted [2, 6], traditional solutions that optimize only the software or only the hardware can no longer deliver large benefits in computing performance and efficiency. The previous courses described above are focused mainly on the hardware architecture of multi-core systems. Extending them to include software for multi-core systems would stretch the material too much and will likely make each course shallow. To provide a deeper understanding of both hardware and software issues in multi-core systems to the students, we are planning to develop a next-level course that is solely focused on multi-core systems, assuming that the student has taken both 18-741 and 18-742. The course will take a strong interdisciplinary approach: it will focus on the interactions between software and hardware in especially the memory system and on-chip interconnection networks of multi-core systems in each lecture. The purpose is not to develop a yet-another seminar course: there will be regular lectures that discuss both recent research and existing implementations, and how they can be adapted to a hardware/software co-design. Students will propose and perform interdisciplinary projects involving changes to both hardware and software. The end goal is to produce versatile and focused computer architecture PhD students who can think out of the box and are capable in advancing the state-of-the-art by enabling cooperation between hardware and software.

IV. Synergistic activities for university-wide multi-core education and awareness: These activities are described in detail above. We believe incorporating multi-core concepts to other courses and educating the university community on multi-core are longer-term activities we will start within the context of this proposal, but they will last for years. The author has already started some of these activities: he has been holding conversations with faculty members and has given an ECE Seminar on one possible security problem that arises with the arrival of multi-core systems [3].

Expected Outcomes: The longer-term (4+ years) outcomes this proposal is intended to serve are 1) the establishment Carnegie Mellon as a major center in parallel computing education and research and 2) the attraction and formation of large industry-supported parallel computing education/research centers, similar to the Universal Parallel Computing Research Centers formed at UC-Berkeley and University of Illinois by Microsoft and Intel, and the Multi-Core Center at UPC-Barcelona, supported by Microsoft. The medium-term (1-2+ years) outcomes are 1) the education of students that are highly-seeked for employment in parallel computing, 2) the production of cutting-edge and top-quality research in multi-core systems, 3) the enabling of effective use of multi-core technology in other departments and on campus. The short-term and immediate outcomes are 1) a revised and modernized computer architecture curriculum that is adapted to trends in industry and technology and 2) attraction of high-quality students who are interested in making a difference in parallel computing hardware/software.

Evaluation: The ultimate evaluation of the proposed activities will come from the success of students who are educated with the revised curriculum. Measuring this success is a long-term process, which cannot be evaluated within the timeline of this proposal. Instead, we intend to collect immediate feedback from faculty, students, and industrial partners to evaluate our success. In particular, we will use the perceived amount of learning (as described by students and faculty) and the perceived level of competence (as described by industrial partners and colleagues who recruit our graduates) in multi-core systems as measures for evaluation. In addition, we will measure the quality and number of published research articles that stemmed from the revised curriculum (and course projects) as a measure of evaluation of the success of the program.

Timeline: Figure 2 shows the timeline of major milestones. Activities colored in green have already been started.

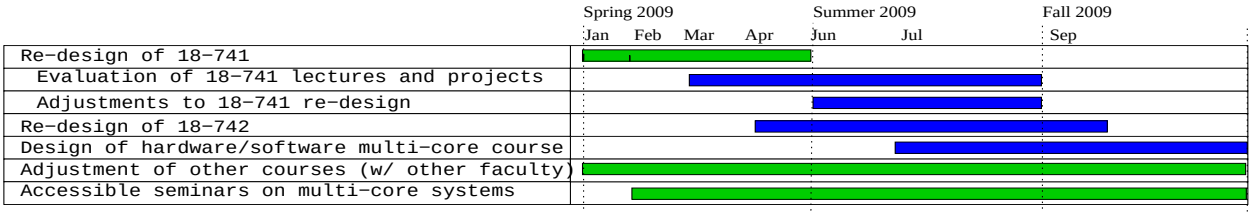

Figure 2. Major milestones and expected completion dates

References

[1] S. Borkar. Thousand core chips: A technology perspective. In *Design Automation Conference*, 2007.

[2] W. W. Hwu et al. Implicitly parallel programming models for thousand-core microprocessors. In *Design Automation Conference*, 2007.

[3] O. Mutlu. Preventing Memory Performance Attacks in Multi-Core Systems. CMU ECE Seminar, Feb. 2009.

[4] O. Mutlu and T. Moscibroda. Parallelism-aware batch scheduling: Enhancing both performance and fairness of shared DRAM systems. In *35th International Symposium on Computer Architecture*, 2008.

[5] M. K. Qureshi, D. N. Lynch, O. Mutlu, and Y. N. Patt. A case for MLP-aware cache replacement. In *33rd International Symposium on Computer Architecture*, 2006.

[6] B. Smith. *Reinventing Computing*. Microsoft Technical Fellow. Keynote at the 2006 International Supercomputing Conference.