Teaching Statement

John Wawrzynek
July 2013

I have focused my teaching on design courses, an area in which I believe I have much to offer. Electronic design, not only involves quantitative analysis (trading off cost, performance, and energy efficiency), but also brings in a strong element of creativity and at times is more like art than science. It gives me great pleasure to teach students basic design skills and to help them call on their (sometimes yet untapped) inner creativity to move from concept to physical realization. Many students have told me that their experience in one of my design courses has been a defining moment of their education at Berkeley.

CS150: Digital Systems Design

My main undergraduate course has been CS150, Digital Systems Design. Since Fall of 1998, I taught the course 10 times.

Starting in 2008 I began a complete revamp of the equipment and content of the CS150 Digital Design Laboratory. CS150 has a major laboratory and project component, and the success of the course over the years has been largely due to our periodic introduction of state-of-the-art equipment. Students had been using development boards that we had introduced more than 5 years earlier.

I led a small study to evaluate commercially available FPGA development boards. After choosing the Xilinx XUPV5-LX110T development boards, I negotiated a donation for 50 boards for the CS150/152 laboratory (a $37,500 value, based on regular academic pricing). I led a team to develop new software and design infrastructure for continued use in the lab, based on the new boards. I also devised and led the implementation of all new laboratory exercises using the new software and hardware. The new lab content was rolled out in Spring of 2009, with much success.

Since then I have continued to refine and evolve the course content. The most significant changes over the years has been 1) a shift from low-level hand optimization techniques (such as Karnaugh maps) to a higher reliance on logic synthesis, and 2) a de-emphasis of Boolean Algebra and introduction of higher-level system organization principles. My lecture notes and laboratory materials are now the standard set used by the other course instructors.

CS61C: Machine Structures

Since Fall of 1998 I taught CS61c, Machines Structures, four times (most recently in Fall 2007). Prior to Spring 2006, I organized a coordinated effort among the instructors of our undergraduate computer architecture and hardware courses to move material from CS150 into CS61C. I oversaw the change of the course’s unit value from 3 to 4, introducing new material on digital synchronous circuits and systems and created new lecture material, laboratory exercises, homework, and projects, still in use today. This change has had the net positive effect of rounding out the CS61C student experience and making them better prepared for CS150, CS152, and other upper division courses.

CS250: Digital Design

Since Fall of 1998, I’ve taught CS250, VLSI Design, 6 times.

When I joined the faculty in 1988, the course had followed the Mead/Conway model. The vision for the course was to teach chip design skills to those in the best position to use them—system designers. A hallmark of the course then was a large student chip design project. However, with the evolution
of Moore’s Law, digital chips became larger and exponentially more complex, and it became increas-
ingly difficult to complete a significant or unique chip design project in a single semester. Throughout
academia, simulation and FPGA emulation eventually replaced hands-on chip design. An unintended
consequence of this change was that computer system architecture graduate students didn’t have the
skills needed to build chips. However, system architects need to understand and be skillful in low-level
implementation tactics to get meaningful results, as optimizations cross many levels from representa-
tion from algorithm through physical implementation. Emulation and simulation provide insufficiently
accurate models of energy consumption, performance, and cost (chip area). Besides, reducing a design
idea to silicon is a strong motivator for students—something missing with emulation approaches.

In Fall of 2009, Krste Asanovic and I worked to remedy this situation by restoring CS250 to a strong
hands-on design course. We had been working on a new hardware description language with associated
tools and library components for chip design, called Chisel. The advances that Chisel makes in ease of
design gave us the opportunity of reintroduce a significant chip design into the course and have students
finish within one semester.

In the course, thought the lectures, we teach the basics of digital circuit to the appropriate level
without going too far down the EE path. Meanwhile the students complete lab exercises designed to
teach them the basics of using the design tools.

The majority of the semester is spent on student projects. Several factors make this part of the
course unique. First, we treat the project development as a workshop, with the teaching staff meeting
weekly with each project group giving them detailed critiques and suggestions, and students making
several presentations to the entire class. Both these help keep the project groups focussed and helps
them to complete an aggressive design and analysis by the end of the semester.

The other unique thing about the project portion is that students not only complete a single design,
they complete a series of designs meant to explore the design space of some particular function—
trading off chip area (cost) for performance, or energy efficiency for cost. The objective is for students
to explore the Pareto optimal frontier for VLSI implementation their function.

I have taught the revived CS250 course along this model once per year for four years now—three
times with Asanovic and once with Jonathan Bachrach. Measured by the number and quality of com-
pleted projects and research project spin-offs, we have been successful.

We have developed the materials and course model to the point now where others can teach it
without either Krste or myself. Jonathan Bachrach is scheduled to teach the course in Fall 2013.

CS294-59, Fall 2010: The Technology and Business of FPGAs
In this course we took a detailed look at the technology and business of FPGAs—what circuits and
chip architectures have worked, which ones haven’t, and why. In the 25 years since their invention,
field programmable gates arrays (FPGAs) evolved from simple logic devices, with a few tens of logic
devices, to full-fledged programmable systems-on-a-chip, with hundreds of thousands of logic cells,
hundreds of dedicated arithmetic blocks, and embedded microprocessors. Consequently, the market
for FPGAs has evolved from simple random logic replacement to high-performance applications in a
variety of areas such as system prototyping, networking, signal-processing, and scientific computing.
Incredibly, during this same period, more than 50 startup ventures came and went trying to capitalize
on the growing demand for programmable logic.

CS298-48, Fall 2009: Hardware Design Patterns, with K. Asanovic
This course was a graduate seminar in which we will attempted to construct a hardware pattern language to capture common solutions to digital hardware engineering problems. The hardware pattern language included common techniques for structuring computation, communication, and synchronization in the context of large-scale digital VLSI systems. The intent was for the pattern language to be later used as a conceptual framework for teaching design and for developing high-level hardware design tools.

**CS294-3, Spring 2004: Reconfigurable Computing**

The primary objective of work in reconfigurable computing is to develop a better alternative to the computing model followed in traditional processors. In this course we covered the basic foundations of computing with FPGAs and other reconfigurable devices, surveying the successes (and failures) in architectures and mapping tools, and a wide range of applications. The objective of the course was to form an understanding both how to architect reconfigurable systems and how to apply them to solving challenging computational problems. The course was structured as lectures from the instructor and visitors used to set the stage for conference paper reading and class discussion, along with independent work on projects.

**Outside Teaching**

In May of 2009 I was appointed Guest Professor of the National Innovation Education Model for Software and Embedded Systems at the Beijing University of Technology, for a period of 3 years. I began this appointment with the keynote talk at the opening of the National Embedded Systems center, Beijing University of Technology (BJUT) FPGAs in Education at UC Berkeley, 5/22/2009.