A group of us at a half-dozen universities [have joined] forces around a common hardware design that connects 40 FPGAs together. We plan to develop a 1,000-way multiprocessor based on standard instruction sets that can run standard software stacks. We would then share that "gateware" and software with the research community.

than to see your computer run real programs successfully for the first time.

In addition to FPGAs, we would leverage the small open source movement for hardware (see opencores.org). They offer big hardware blocks that work, like processors, Ethernet controllers, and so on that can be used for either real chips or FPGAs. Thus, the processor is the lowest-level building block, taking over from the transistor or nand gate.

The growth of FPGA resources plus the open source hardware movement creates the new opportunity. We can place 25 pipelined-processors inside a single large FPGA today, and the number of processors should double every 18 months. By starting with working processors as the building blocks, students can tackle interesting issues facing the field today, such as how to design computer systems to make it easier to write parallel programs or memory architectures for high-performance garbage collection. I'd love to design a supercomputer in such a course.

This technological opportunity has inspired a group of us at a half-dozen universities to join forces around a common hardware design that connects 40 FPGAs together. We plan to develop a 1,000-way multiprocessor based on standard instruction sets that can run standard software stacks. We would then share that "gateware" and software with the research community. We believe it will be an attractive platform for ramping up hardware and software research in multiple processors, and that it can be economically replicated so that many departments could have one. This vision gave the project its name: Research Accelerator for Multiple Processors, or RAMP [2]. When completed, it will enable students to explore and share even grander designs than a single FPGA. (To learn more, see ramp.eecs.berkeley.edu/.)

Conclusion

Our computer science curriculum was developed at a time when it was the first introduction students had to computing and the software we use everyday had not yet been written. There have been extraordinary developments since then, and most students arrive on campus today with years of computer experience. While one challenge to our curriculum is to catch up to the technology of the 21st century, another is we haven't leveraged the better background of students. A third challenge is for CS faculty to learn new ways.

Let's learn and try creating CS courses that we'd love to take and teach, and that will capture the exciting opportunities and challenges of our field and of our students. ■

References

1. Anastassia A. and Hellerstein, J.M. Exposing undergraduate students to database system internals. *ACM SIGMOD Record 32*, 3 (Sept. 2003), 18–20.
2. Arvind, K.A., Chiou, D., Hoe, J.C. Kozyrakis, C., Lu, S-L, Oskin, M., Patterson, D., Rabaey, J., and Wawrzynek, J. *RAMP: Research Accelerator for Multiple Processors—A Community Vision for a Shared Experimental Parallel HW/SW Platform.* Technical Report UCB//CSD-05-1412, University of California, Berkeley, Sept. 2005.
3. Hennessy, J. and Patterson, D. *Computer Architecture: A Quantitative Approach, 4th Edition* (to appear Oct. 2006).
4. NSF Workshop. Interative Computing Education and Research—Preparing Graduates for 2010 and Beyond. (Jan. 27–28, Palo Alto, CA); www.evergreen.edu/icer.

These ideas came from conversations with many friends and colleagues, especially Nina Bhatti of Hewlett-Packard Labs, James Gosling of Sun Microsystems, Joe Hellerstein of UC Berkeley, John Hennessy of Stanford, Jim Larus of Microsoft Research, and Kathy Yelick of UC Berkeley.

David A. Patterson (pattrsn@cs.berkeley.edu) is president of ACM and the Pardee Professor of Computer Science at the University of California at Berkeley.