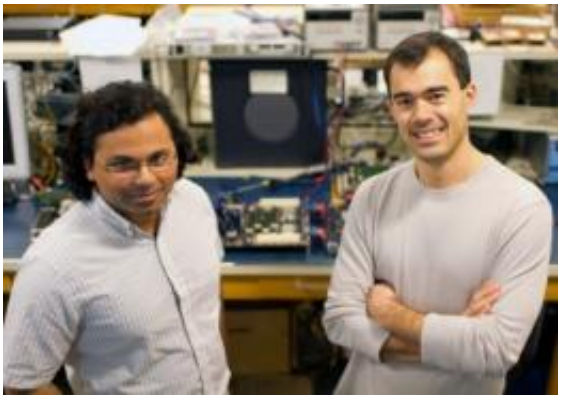
Streamlining chip design

December 8 2011, by Larry Hardesty



Nirav Dave PhD '11, left, and Myron King. Photo: Melanie Gonick

In the same way that computing power moved from mainframes to the desktop in the 1980s, it's now moving from the desktop to handheld devices. But that's putting new demands on chip designers. Because handhelds are battery powered, energy conservation is at a premium, and many routine tasks that would be handled by software in a PC are instead delegated to special-purpose processors that do just one thing very efficiently. At the same time, handhelds are now so versatile that not everything can be hardwired: Some functions have to be left to software.

A hardware designer creating a new device needs to decide early on which functions will be handled in hardware and which in software. Halfway through the design process, however, it may become clear that something allocated to hardware would run much better in software, or

vice versa. At that point, the designer has two choices: Either incur the expense — including time delays — of revising the design midstream, or charge to market with a flawed device.

At the Association for Computing Machinery's 17th International Conference on Architectural Support for Programming Languages and Operating Systems, researchers from MIT's Computer Science and Artificial Intelligence Laboratory (CSAIL) will present a new system that enables hardware designers to specify, in a single programming language, all the functions they want a device to perform. They can thereafter designate which functions should run in hardware and which in software, and the system will automatically churn out the corresponding circuit descriptions and computer code. Revise the designations, and the circuits and code are revised as well. The system also determines how to connect the special-purpose hardware and the general-purpose processor that runs the software, and it alerts designers if they try to implement in hardware a function that will work only in software, or vice versa.

The new system is an extension of the chip-design language BlueSpec, whose theoretical foundations were laid in the 1990s and early 2000s by MIT computer scientist Arvind, the Charles W. and Jennifer C. Johnson Professor of Electrical Engineering and Computer Science, and his students. BlueSpec Inc., a company that Arvind co-founded in 2003, turned that theoretical work into working, commercial code.

As Arvind explains, in the early 1980s, an engineer designing a new chip would begin by drawing pictures of circuit layouts. "People said, 'This is crazy,'" Arvind says. "'Why can't I write this description textually?'" And indeed, 1984 saw the first iteration of Verilog, a language that lets designers describe the components of a chip and automatically converts those descriptions into a circuit diagram.

BlueSpec, in turn, offers an even higher level of abstraction. Instead of describing circuitry, the designer specifies a set of rules that the chip must follow, and BlueSpec converts those specifications into Verilog code. For many designers, this turns out to be much more efficient than worrying about the low-level details of the circuit layout from the outset. Moreover, BlueSpec can often find shortcuts that a human engineer might overlook, using significantly fewer circuit components to implement a given set of rules, and it can guarantee that the resulting chip will actually do what it's intended to do.

For the new paper, Arvind, his PhD student Myron King, and former graduate student Nirav Dave (now a computer scientist at SRI International) expanded the BlueSpec instruction set so that it can describe more elaborate operations that are possible only in software. They also introduced an annotation scheme, so the programmer can indicate which functions will be implemented in hardware and which in software, and they developed a new compiler that translates the functions allocated to hardware into Verilog and those allocated to software into C++ code.

Today, King says, "if I consider my algorithm just to be a bunch of modules that I've hooked together somehow, and I want to move one of these modules into hardware, I actually have to re-implement it. I have to write it again in a different language. What we're trying to give people is a language where they can describe the algorithm once and then play around with how the algorithm is partitioned."

King acknowledges that BlueSpec's semantics — describing an algorithm as a set of rules rather than as a sequence of instructions — "is a radical departure from the way that most people think about software." And indeed, among chip designers, Verilog is still much more popular than BlueSpec. "But it's precisely this way of thinking about computation that allows you to generate both hardware and software," King says.

Rajesh Gupta, the Qualcomm Professor in Embedded Microsystems at the University of California at San Diego, who wasn't involved in the research, agrees. "Oftentimes, you need a dramatic change, not for the sake of the change, but because the problem demands it," Gupta says. But, he adds, "hardware design is hard to begin with, and if some group of very smart people at MIT — who are not exactly known for making things simple — comes up with what looks like a very sophisticated model, some people will say, 'My chances of making a mistake here are so high that I better not risk it.' And hardware designers tend to be a little bit more conservative, anyway. So that's why the adoption faces challenges."

Still, Gupta says, the ability to redraw the partition between hardware and software could be enticing enough to overcome hardware designers' conservatism. If you're designing hardware for portable devices, "you need to be more power efficient than you are today," Gupta says. But, he says, a device that relies too heavily on software requires so many layers of interpretation between the code and the circuitry that "by the time it actually does anything useful, it has done many other things that are useless, which are infrastructural." To design systems that avoid such unnecessary, energy-intensive work, "you need this integrated view of hardware and software," he says.

Provided by Massachusetts Institute of Technology