Configurable Chips Meld Software and Hardware

Moshe Sipper and Eduardo Sanchez
Swiss Federal Institute of Technology, Lausanne

What would happen if you tossed a software designer and a hardware engineer into the same room and threw away the key? Once upon a time, you might expect to see smoke seeping out from under the door as the two engaged in a fiery debate. Today, such an argument would more likely require the services of a psychiatrist than a mediator, for, increasingly, a single person practices both professions.

Nearly 10 years ago, Maurice Wilkes ("It's All Software, Now," *Comm. ACM*, Oct. 1990, pp. 19-21) observed that IC design had become mostly a software affair. He wrote that "Formerly, a circuit designer needed, when checking out a design, the practical skills associated with working at a laboratory bench. Now, certain software skills are necessary instead. The designer needs to be comfortable working with large software systems, and must know how to fight the system when that is necessary." He then predicted, "It seems likely that, in the future, all circuit designers will have a strong software background."

FROM PROGRAMMABLE TO CONFIGURABLE CHIPS

It seems Wilkes' prophecy has been fulfilled, thanks to the maturation of field-programmable gate arrays and to recent advances in hardware synthesis tools—developments that have given rise to the new configurable-computing paradigm (J. Villasenor and W.H. Mangione-Smith, "Configurable Computing," *Scientific American*, June 1997, pp. 54-59).

Large and fast FPGAs, field-programmable gate arrays can be modified to perform figured at almost any point by the end user. FPGA technology brings about a primary distinction between *programmable* and *configurable* processors (Eduardo Sanchez et al., "Static and Dynamic Configurable Systems," *IEEE Trans. Computers*, June 1999, pp. 556-564). With programmable, general-purpose processors, you can change the software at any given moment via programming—but the hardware remains immutable once the processor leaves the foundry. Configurable processors, on the other hand, remain mutable at the hardware level: The elemental logic gates, interconnections, inputs, and outputs can be programmed and reprogrammed—configured and reconfigured—by the end user, in the field. FPGAs thus combine the prime advantage of general-purpose computers, programmability, with the prime advantage of application-specific integrated circuits, efficiency.

FPGAs now occupy a rather small niche in the computing industry; engineers are using them in applications like image processing, data encryption, and bio-inspired hardware, which make use of an FPGA's reconfigurability. Last year, one company announced a general-purpose microprocessor chip, 65 percent of whose surface is reconfigurable, with only 35 percent serving as a classic controller to manage the chip's operation (J. Turley, "Triscend E5 Reconfigures Microcontrollers," *Microprocessor Report*, Nov. 16, 1998, pp. 12-13). This trend will continue, with general-purpose processors incorporating more reconfigurable, on-chip surface. In a few years such hybrid programmable and configurable processors could play a major role within the computing industry.

MAKING HARDWARE SOFT

Programming either FPGAs or hybrid processors usually involves software.

Processors that can be configured by end users promise to combine hardware's speed and efficiency with software's flexibility.

Today, the same trend that Wilkes observed over a decade ago has expanded to encompass a much larger scale: Configurable processors work hand in hand with a medley of software tools, and the typical hardware designer now relies more on a workstation than an oscilloscope. Hardware engineers thus find themselves increasingly using software tools, while a growing number of software designers have begun using configurable circuits.

Soft logic

Logic design—specifically the design of digital circuits—while not necessarily less hard today than in years past, is at least less hardware-intensive. For example, suppose you want to develop a

device that recognizes English characters. You could do so by writing a program in a high-level language such as C++ or Java, with the intent of running it on a standard PC or workstation.

Alternatively, you could take the hardware approach and use a configurable circuit and a high-level hardware description language, such as VHDL, to write your character-recognition program. Because the language you're using operates at a high level, you can design your circuit from the top down, choosing the level of abstraction at which you feel comfortable working. Doing so lets you use the hardware description language to shield you from most hardware design drudgery.

For example, with such software tools, *schematic capture*—the design of the circuit layout and interconnects—occurs automatically, generating the circuit design using a compiler that processes a high-level description. Yet circuit designers can still intervene directly from time to time to manually optimize the layout. Moreover, developers can now purchase intellectual-property hardware modules. These IP modules echo the traditional software packages, in which not everything is programmed from scratch, but consists in part of prebuilt software modules, known as packages or libraries. The hardware approach, however, offers a faster and more efficient character-recognition machine than the programmed-workstation version could provide.

Codesign codependencies

Logic designers who use such high-level software tools must, however, confront a new problem—*codesign*, which addresses the joint development of a system's software and hardware components. For example, today's technology would not permit our character-recognition machine to be implemented *entirely* in configurable hardware, because doing so would cost too much. To save money, we would still rely on software for some parts of our machine. How would the designer determine where to place this dividing line between software and hardware? Generally, an experienced codesigner would attempt to implement an application's time-consuming components in hardware, thus maximizing execution speed. Known as the *partitioning problem*, this issue is but one of several that confront codesigners.

Resources

For information on the **International Workshop on Hardware/Software Codesign (CODES)** see http://www.computer.org/conferen/proceed/codes/8442/8442toc.htm.

Two sites offer information on the **IEEE Symposium on FPGAs for Custom Computing Machines (FCCM)**, another international conference in the field of hardware-software codesign: http://www.fccm.org and http://www.computer.org/conferen/proceed/fccm/8900/8900toc.htm.

For more information on *Labomat* (from *laboratoire matériel*, French for hardware laboratory), see http://lslwww.epfl.ch/labomat.

Developments in configurable computing increasingly blur the line between hardware and software, a trend that represents a major shift in computing practice. To keep their offerings current and relevant, universities should modify their computer science curricula to better prepare students for this new era. Although hardware design is much more software-oriented now, aspiring computing students still need courses that cover hardware synthesis techniques, codesign methodologies, and module reuse strategies. Students should also experience working in teams.

At our institute, we have developed such a new curriculum. By offering several hardware courses at both the elementary and advanced levels, we help students obtain a deeper and broader knowledge of configurable hardware design. Moreover, over the past few years we have invested in the design and construction of several FPGA-based boards for use as teaching platforms. These boards let students gain hands-on experience with technologies they'll use in real-world jobs. For example, using our *Labomat* board, a three-student team designed and tested a simplified floating-point unit in five three-hour sessions. ✷

Moshe Sipper is a senior researcher and Eduardo Sanchez is a professor with the Logic Systems Laboratory at the Swiss Federal Institute of Technology, Lausanne. Contact Sipper at moshe.sipper@epfl.ch and Sanchez at eduardo.sanchez@epfl.ch.

Editors: Jerzy W. Rozenblit, University of Arizona, ECE 320E, Tucson, AZ 85721; jr@ece.arizona.edu; and Sanjaya Kumar, Honeywell Technology Center, MS MN65-2200, 3660 Technology Dr., Minneapolis, MN 55418; skumar@htc.honeywell.com.