Will hardware and software be codesigned?

Grant Martin
Tensilica

The dream of true hardware-software codesign of complex electronic products has been in people’s minds for many years. A quick Google search will find references going back to the early 1990s, and many of the system-level design approaches and research of the 1980s assumed some flexibility in the interface between hardware and software. The international symposium on hardware/software codesign (CODES) started in 1991, and work presented at this series of conferences has defined much of the field.

What is hardware-software codesign? There are as many definitions of this as there are people working in the field. We can’t do better than the two definitions in chapter 1 of *A Practical Introduction to Hardware/Software Codesign*; the first definition being the traditional one:

Hardware/Software codesign is the design of cooperating hardware components and software components in a single design effort.

After a discussion of the many forms of hardware and software in use in design today, Schaumont proposes a more refined and subtle definition:

Hardware/Software codesign is the partitioning and design of an application in terms of fixed and flexible components.

Why did hardware-software codesign emerge? I like to think of it as a magic confluence: designers grew frustrated with the assumption that the world needed to be inevitably partitioned into the old computing paradigm of big mainframes with fixed instruction sets and the clunky software that ran on them as, simultaneously, new implementation technologies arose giving much more freedom to designers to design systems that drew different boundaries than the old paradigms offered. A Kuhnian scientific revolution arose when early ASICs, early field-programmable devices, and application-specific instruction-set processors emerged and taught a new generation of designers that the old models and paradigms could be ground into the dust.

But despite the fact that many designers participated in these revolutions, the new ways of thinking have quickly hardened into traditional methods. Educating each generation of designers to understand that the boundaries between hardware and software in their systems can and should be fluid requires a constant effort to find ways to teach students that there are ways to relax the traditional constraints. In a crowded engineering or computing curriculum, finding the time for the traditional curriculum is a challenge, let alone the new subjects they should learn.

Into this gap comes the new book by Patrick Schaumont, professor at Virginia Tech, *A Practical Introduction to Hardware/Software Codesign*. As I read it, I noted four very significant features of the book, which can be a textbook for courses on hardware-software codesign or just read by anyone as a structured introduction to the topic:

1. It is practical. It is stuffed with code samples, diagrams, outputs from tools, listings, and all manner of useful artifacts.
2. It is introductory. Written as a textbook, it is based on Schaumont’s experience in developing and teaching courses on hardware-software codesign to upper-year undergraduates and beginning graduate students in electronic engineering and computing.

3. It contains Schaumont’s personal approach to hardware-software codesign. This is a structured build, from the fundamentals of modeling through the range of implementation choices through system-level interfaces and, finally, examples.

4. It is well written and easy to read. In particular, the knitting together of examples, code listings, outputs from tools—and the explanations of all these—is very well done.

The book is structured in four parts. The first part, “Basic Concepts,” discusses first the nature of hardware-software codesign and defines it as we first mentioned. Two more chapters deal with fundamentals of modeling systems in ways that are appropriate to hardware-software implementation analysis and trade-offs: dataflow modeling, and control and dataflow analysis.

This is a natural lead-in to Part 2, “The Design Space of Custom Architectures,” which surveys a variety of implementation approaches that are relevant to the flexible design of systems. Starting with FSMD (finite state machine with datapath), the chapters move from more hardware-oriented building blocks to more software-oriented processors and subsystems. Chapter 5, on microprogrammed architectures, is a good reminder of design approaches that peaked in use in the past yet are still relevant to understanding today’s choices. Chapter 6 introduces general-purpose embedded processor cores with a focus on RISC, and chapter 7 concludes this part with a discussion of SoC architectural choices.

Part 3 of the book, “Hardware/Software Interfaces,” contains three chapters that introduce quite a mixture of topics and which reflect several areas of SoC and subsystem design that readers need to know. Chapter 8 deals with on-chip buses, using three real examples: AMBA, CoreConnect, and Wishbone. At the end of the chapter, there is a brief discussion of the network-on-chip (NoC) concept, appropriate to introduce here as on-chip buses are evolving to embrace more NoC principles. Chapter 9, on hardware-software interfaces, discusses topics in a true integrating style: memory-mapped interfaces with a focus on synchronization approaches; coprocessor interfaces with examples; and finally, something that is somewhat surprising in this chapter, a discussion of application-specific instruction set processors (ASIPs). This discussion, all too brief, is a useful way to expose the ASIP approach to codesign to students—something that many of them may not have heard of before using this book. Chapter 10 dives into detail on coprocessor control shell design.

The final part of the book contains two detailed application descriptions: a coprocessor for streaming encryption, and a coprocessor for evaluating

### ADVERTISER INFORMATION • MARCH/APRIL 2011

#### Advertising Personnel
Marian Anderson: Sr. Advertising Coordinator
Email: manderson@computer.org
Phone: +1 714 821 8380 | Fax: +1 714 821 4010

Email: sbrown@computer.org
Phone: +1 714 821 8380 | Fax: +1 714 821 4010

IEEE Computer Society
10662 Los Vaqueros Circle
Los Alamitos, CA 90720 USA
www.computer.org

#### Advertising Sales Representatives
Western US/Pacific/Far East: Eric Kincaid
Email: e.kincaid@computer.org
Phone: +1 214 673 3742
Fax: +1 888 886 8399

Eastern US/Europe/Middle East: Ann & David Schissler
Email: a.schissler@computer.org, d.schissler@computer.org
Phone: +1 508 394 4026
Fax: +1 508 394 4926

#### Advertising Sales Representatives (Classified Line/Jobs Board)
Greg Barbash
Email: g.barbash@computer.org
Phone: +1 914 944 0940
Cordic algorithms. The chapter on coprocessor control shell design is key background for some of the details on coprocessor hardware interfaces and device drivers. In these chapters, different trade-offs for different implementation approaches, including ASIP processor extension, are explored.

There are some explicit choices made in this book that some might question. To illustrate the codesign principles, Schaumont uses a hardware description language of his own devising: GEZEL rather than Verilog, VHDL, or SystemC. He justifies it on the grounds of relative simplicity as a learning and teaching vehicle, and brevity and comprehensibility when compared to the other choices. In addition, GEZEL and supporting tools are available for download, and indeed, GEZEL can be automatically translated to VHDL for implementation experiments. In chapter 4, an example is used to demonstrate the brevity and simplicity of GEZEL vs. the other HDLs. Since it is more important for students to learn principles of design and languages than a particular syntax, and since Schaumont backs it up with tools and the proof of successful course teaching, I think his decision here is a right one.

This is not a book for theoreticians. The first part deals with modeling theory to some extent in discussing data and control flow models, but the text is written from a “need-to-know” perspective, and the reference list for the book is small and carefully chosen so as not to overwhelm students. Perhaps a two-part reference list would have been useful with the selected reading list first, followed by a more extensive set of references for those wanting to delve much deeper.

**Finally, coming from** the extensible and configurable processor IP side of the industry, I wish that even more discussion of this configurable ASIP approach was in the book. But that of course is a natural bias of mine!

These are small differences of emphasis on what is an extremely useful text on codesign. If I were teaching a course on this subject, I would use this as a resource and text. If I were a student who wanted to learn codesign, I would look for a course that at least used a similar approach. If I were an engineer or engineering manager who wanted to learn more about codesign from a very practical perspective, I would read this book first before any other. When I first started learning about codesign as a practitioner, a book like this would have been the perfect introduction. The design industry will benefit from having more people who understand what codesign is and where it can be applied.

Direct questions and comments about this department to Grant Martin, 2424 Raven Road, Pleasanton, CA 94566; gmartin@ieee.org.

Selected CS articles and columns are also available for free at http://ComputingNow.computer.org.
Running in Circles
Looking for a Great Computer Job or Hire?

Make the Connection - IEEE Computer Society Jobs is the best niche employment source for computer science and engineering jobs, with hundreds of jobs viewed by thousands of the finest scientists each month - in Computer magazine and/or online!

IEEE Computer Society | JOBS
http://www.computer.org/jobs

IEEE Computer Society Jobs is part of the Physics Today Career Network, a niche job board network for the physical sciences and engineering disciplines. Jobs and resumes are shared with four partner job boards - Physics Today Jobs and the American Association of Physics Teachers (AAAPT), American Physical Society (APS), and AVS Science and Technology of Materials, Interfaces and Processing Career Centers.