HARDWARE/SOFTWARE CO-DESIGN: A SHORT COURSE FOR UNBELIEVERS

A. C. Downton, M. Fleury, R. P. Self, S. J. Sangwine and P. D. Noakes

Department of Electronic Systems Engineering, University of Essex, UK.
email: acd@essex.ac.uk

Abstract

Hardware-Software Co-Design skills are increasingly needed to implement complex distributed embedded systems and systems-on-a-chip for applications in Telecommunications and related fields. Telecommunications and network engineers and computer scientists frequently lack the electronic engineering background to be able to design and implement such systems. Our short course, presented as an option within two different specialist postgraduate degree programmes, and also available direct to industry, has successfully demonstrated that foundation hardware-software co-design skills can be acquired in 2 days by students with limited previous hardware background.

1 Introduction

The Department of Electronic Systems Engineering at the University of Essex specialises in Telecommunications and Computer Networks in its postgraduate teaching (see [1]). Its MSc core courses are augmented in the Spring Term by a programme of specialist options, which are presented in short course format so as to be accessible also to practising engineers in the telecommunications industry. Both the full MSc and the industry short courses normally attract graduates with a background in telecommunication engineering, computer networks or, increasingly, in computer science. Although the department also has undergraduate courses and research interests in ECAD, it has not previously been possible to integrate these with our MSc programmes because relevant material on analogue and digital IC and system design was not close enough to the core teaching needs and prior background of our students.

Recent developments in the telecommunications industry however, both in consumer equipment and service provision, are increasingly focusing on embedding intelligence within equipment and systems, and integrate many traditional areas of electronics into a minimum number of components. As the complexity of these systems is also continuously increasing, the only way for designs to be completed cost-effectively is to reduce the range of electronics skills required by the design team by design automation. Hardware-Software Co-Design attempts to achieve this by developing an integrated design path, based around a combination of software development and semi-configurable hardware components. In our case, we have chosen a Hardware-Software Co-Design route at the extreme software end of the spectrum, since we believe that in the longer term, this will become an increasingly dominant area of demand from graduates.

2 Hardware-Software Co-Design

Hardware-software co-design is an area which is of major and increasing interest to the telecommunications industry. Embedded systems including mobile handsets, base stations, digital radio, set-top boxes and core network infrastructure utilise complex combinations of computing, DSP, analogue and digital components. Increasingly, such components are integrated into ASICs, multi-processor embedded systems or systems-on-a-chip requiring extensive design skills that would traditionally involve a major design team covering everything from telecommunications to computer engineering to ECAD design. With declining numbers of electronics engineering students and increasing numbers of computing/networks students, there is a significant shortage of graduates with suitable backgrounds to engage in detailed chip design for telecommunications systems. Designing a two-day short course which provides an accessible introduction to hardware/software co-design for graduates with limited or no background in traditional ECAD, yet enabling them to reach a point where they can return to industry with sufficient knowledge and awareness to be productive in this

field within a few months, presents a significant challenge. In our particular case, the course was also taken by a number of MSc students from our Computer Science department, undertaking a Masters-level Training Programme in Robotics and Embedded Systems [2], typically with prior background of a computer science degree.

3 Short Course Structure

An obvious danger in designing a two-day short course is that, within the space of about 10 lecture hours, a number of topics will be superficially covered, but there will be insufficient time to address any topic in the depth required for students to gain a proper insight into the subject area. We chose to address this problem by designing our course around a core of using Handel-C [3] with Xilinx platform FPGAs [4] as a suitable fast-track route for computer and network engineers to approach hardware-software co-design. To ensure that students engaged fully with the lecture material, we further reduced the core lectures to 8 hours, but augmented this with a seminar and a hands-on assignment (using the Celoxica DK1 design suite and RC100 FPGA prototyping boards). Our overall course structure is shown in Figure 1. For MSc students the initial two-day short course was augmented by further time spent in the laboratory completing the self-paced assignment (which contributed 15% of overall course marks), and by revision seminars held later in the academic year.

Day	Time	Activity
Day 1	09.00-10.00	Registration
	10.00-11.00	Introduction to Co-Design
	11.30-12.30	Introduction to Handel-C
	13.30.14.30	FPGA Hardware Compilation
	14.30-15.30	Handel-C Software to Hardware (seminar and demonstration)
	16.00-17.00	Handel-C hands-on laboratory session
	17.00-18.00	
Day 2	09.00-10.00	High-level Design Models
	10.00-11.00	Hardware Design Languages and VHDL
	11.30-12.30	Design Methodologies
	13.30.14.30	Systems-on-a-chip
	14.30-15.30	Designing with Handel-C
	16.00-17.00	Handel-C hands-on laboratory session
	17.00-18.00	

Figure 1 Essex 2-day Co-Design short course timetable

3.1 Handel-C core

Our course starts with a lecture that introduces and explains the role of hardware-software co-design by reviewing:

- Electronic design industry trends and design complexity issues
- Emerging design productivity and skills gaps
- Why new design paradigms are needed for large-scale distributed embedded systems and systems-on-a-chip
- The current design flow, from abstract specification to structural representation to behavioural domain and low-level EDA automated tools
- How the co-design paradigm fits in, especially for reconfigurable devices
- How Handel-C fits into co-design

Handel-C is then introduced as a variation of conventional C: it is assumed that all students taking the course are already experienced C, C++ or Java programmers. The emphasis of the lecture is in showing the relationship between Handel-C language features and the specific hardware constructs to which they map on the FPGA (described at register rather than signal level). Comparisons are also made with conventional CPU architectures (at a first computer architectures course level). These

highlight the similarity of the FPGA and CPU hardware constructs and the performance benefits which can be gained by unrolling programme loops into data parallel or pipeline form on the FPGA. Emphasis during the lecture is placed upon:

- introducing the novel parallel constructs of Handel-C, which are derived from CSP and Occam [5]; these include PAR and SEQ constructs, channels, and function arrays
- explaining the extended integer data types (which are intended to ensure efficiency of hardware allocation), and the corresponding FPGA memory model.

Handel-C is well-suited to students with limited electronics or CAD background, because extraneous hardware detail (e.g. details of clocks, signals and voltages) are avoided in favour of additional abstract programming concepts, such as concurrency, synchronization, barriers and rendezvous that are generally familiar to computer engineers and scientists. Furthermore Handel-C is (currently) applied on FPGAs which utilise a single global clock, and there is thus an extremely simple relationship between software and clock cycles, viz:

- Each **assignment statement** involves storing the result of the assignment in a (clocked) register and therefore requires 1 clock cycle
- **Expression evaluation** (regardless of the complexity of the expression), is wholly a combinational logic operation and therefore requires zero clock cycles, but results in a propagation delay through the corresponding combinational logic.

The advantage of this simple relationship is that it becomes extremely easy to reason about the number of clock cycles required to execute a piece of code. It also follows however from the use of a single global clock that the overall execution speed (i.e. clock speed) is determined by the worst-case propagation delay resulting from the most complex expression evaluation encountered during silicon compilation. This, in turn, encourages a simplified programming style which maximises compact hardware at the expense of compact code (and is at the opposite end of the spectrum from the obfuscatory programming sometimes favoured by 'real' C programmers!).

3.2 Assignment work

The assignment starts out from a typical 'Hello World' program, which introduces the DK1 development environment (which is similar in style to Visual C++, and therefore readily assimilated by most students). Exercises reinforce the novel concepts of Handel-C programming (i.e. those which differ from conventional C/C++), such as data size optimization for hardware, core parallel constructs such as par statements, function arrays and process mapping to hardware, and register and channel-based pipelines. Most of these concepts are already familiar to students with a reasonable background in computer science; the novelty is in seeing how they map directly to hardware on the FPGA. The concepts are embedded within a series of progressively more sophisticated examples, including an FIR filter and a (very minimal) custom microprocessor implementation, based upon elements of the ARM RISC architecture. In the latter exercises, the students both simulate their Handel-C code using the DK1 simulator within the PC environment of our software teaching laboratory, and build bit-files for the FPGA using Xilinx foundation tools, which are then downloaded and run on RC100s in our research laboratory.

While the initial stages of the assignment take place in scheduled sessions at the end of each day of the short course (and are therefore also available for external industry delegates), the latter stages of the assignment are carried out in self-paced mode to meet a deadline around two weeks after the short course finishes. Our experience was that the two 2-hour initial sessions were sufficient to get students to the point where they could drive the DK1 development system, had completed the first few assignment stages, and had overcome most of the 'finger trouble' problems which often slow down students initial progress. Thereafter, academic support was available via email, and from a postgraduate student demonstrator at the final stage when completed solutions were downloaded and run natively on an FPGA in our research lab.

3.3 Handel-C in context

Only two lectures and one seminar are devoted to introducing core and more advanced concepts of Handel-C (including graphics programming for a VGA video game controller based on an example provided by Celoxica), and a further lecture explains Xilinx foundation tools. The remainder of the course material aims to put Handel-C in context both of the overall system design process and of other design routes for hardware/software co-design. Handel-C is a young language, still rapidly evolving, that derives from a core subset of C with parallel constructs from Occam/CSP grafted on. As such it exhibits both the strengths and weaknesses of its predecessor Occam. Its strength is the provision of a minimal but sufficient set of concurrency constructs (the parsimony principle of Occam's razor) which have simplified the mapping from Handel-C to FPGA gates. Its weakness (indeed, a weakness of parallel computing generally) is the lack of a coherent top-down design methodology which restricts the number of degrees of design freedom that parallelism adds to the conventional waterfall software design process. Another current weakness is the lack of a comprehensive run-time support environment, an active area of our current research.

3.4 Parallel embedded systems design

We address the weakness in design methodology using material developed from our own previous experience in parallel embedded systems design as part of an earlier EPSRC directed research programme on Portable Software Tools for Parallel Architectures. This provides two course lectures. The first overviews a range of current high-level design approaches including:

- modelling languages for large-scale systems
- co-design toolkits for smaller-scale and reconfigurable systems
- data modelling techniques for data-dominated systems
- reactive system modelling for control-dominated systems

The second presents a specific design approach for embedded systems with continuous data [6], and introduces a variety of simple heuristic design techniques based upon Amdahl's Law, pipelining, temporal multiplexing, and data and algorithmic parallelism. This allows students to partition and scale complete sequential algorithms to meet specific throughput and latency specifications using conservative incremental design techniques.

3.5 Broadening material

Finally, it is important to realise the place of Handel-C within the broader spectrum of hardware/software co-design routes. A further lecture therefore provides a more general review of the strengths and weaknesses of other hardware design languages (particularly VHDL) as compared with Handel-C.

Handel-C is a silicon compiler, which attempts to model a programming language in hardware and outputs a netlist compatible with FPGA place-and-route tools. It thus presents a top-down software-oriented model, and matches the background and experience of Computer Scientists well, but provides little insight into the resulting placement and routing on the FPGA chip. This makes analysis of the resulting gate/slice allocation and clock speed somewhat inscrutable. In contrast Hardware Description Languages (HDLs) such as VHDL were generally developed to model hardware using a software language, for use as a CAD tool by Electronic Engineers, and thus present a bottom-up hardware-oriented design model. Their advantages are in their close control of the hardware generated and ability to configure this to meet exact requirements, but at the expense of having to provide considerably greater hardware detail at a level with which many computer scientists will be unfamiliar. Nevertheless, all students should be aware of both approaches and of the relationships between them, which may increasingly result in languages such as Handel-C being used as a form of 'shorthand' substitute for HDLs, even by experienced electronic engineers.

A review of emerging trends towards system-on-a-chip designs completes the course content.

4 Student feedback

Reaction of our students (both MSc and industry) when the course was run for the first time in Spring 2002 was very favourable. The general perception was that the balance between specific detailed material on Handel-C and more general coverage of the field of hardware/software co-design was good, and that typically 80% of the material in the course was new to students, whether their background was as Electronic Engineers, Network Engineers or Computer Scientists - probably about the right balance to ensure effective assimilation. The assignment was clearly highly motivating, as most students spent more time than expected on it and achieved high marks. Perhaps the overall impact of the course can best be summed up in a quote from the feedback form submitted by one of our MSc students: "What I previously had was a theoretical understanding of the whole idea of hardware design which was mostly self read.... My background....in pure computer science didn't offer me any practical opportunities in the area. An interest has been fanned which I hope to develop in the future."

5 Conclusions

Our two-day Hardware-Software Co-Design short course was developed to meet a standard short course format used by the Department of Electronic Systems Engineering at the University of Essex for advanced MSc/industry courses in Telecommunications, Networks and Information Systems. Embedded applications in these areas increasingly make use of hardware-software co-design to build complex distributed embedded systems and systems-on-a-chip. By basing our course around Handel-C rather than traditional HDLs, it has been possible to construct a coherent course of less than 10 lecture hours that was well-received by a broad range of students, included a substantial practical component, and enabled students to achieve significant learning outcomes on the way to becoming competent co-design engineers. The course will be running again in January 2003 (see [7]).

References

1 http://www.essex.ac.uk/ese/msc/index.htm
2 http://cswww.essex.ac.uk/admissions/postgraduate/2001/esr/index.htm
3 Handel-C Language Reference Manual, 3.1, Celoxica Ltd, Didcot, UK, 2001 (http://www.celoxica.com/home.htm)
4 Virtex-II Platform FPGA, Xilinx Inc., San Jose, Calif., 2001 (http://www.xilinx.com)
5 C. A. R Hoare, Communicating Sequential Processes, *Communications of the ACM*, 21(8):666-677, 1978.
6 M. Fleury and A. C. Downton, *Pipelined Processor Farms: structured design for emebedded parallel systems*, John Wiley & Sons, Inc., New York, 2001, ISBN 0-471-38860-2
7 http://www.essex.ac.uk/ese/shortc/index.htm