Bridging Undergraduate Learning and Research in Software and Hardware

Liang Cheng[1] and **Dale Parson**[2]
[1]Laboratory Of Networking Group (LONGLAB, http://long.cse.lehigh.edu)
Department of Computer Science and Engineering, Lehigh University
19 Memorial Drive West, Bethlehem, PA 18015, USA
cheng@cse.lehigh.edu
[2]Agere Systems, Inc.
1110 American Parkway NE
Allentown, PA 18109, USA

Abstract

Embedded processing, where computers are used to monitor and control dedicated hardware, is a growing presence within mainstream computer science and engineering. Network processing, where embedded processors monitor and control communication networks, is a premier example of embedded processing. This paper presents contents of a *Network Systems Design* course used to introduce undergraduate students to understanding software-hardware co-design concepts and acquiring practical experience in embedded processing. The achieved goals of the course include: (*i*) carrying out lab-based introduction to embedded processing in its application area of network processing, and (*ii*) strengthening ties between academic study of network processing and industrial practice in the field, given the fact that most advances in network processor architectures to date have been made in industry. Responses from students approved our intention of the “hands-on” lab-based introduction using a modular network processing laboratory and verified the effectiveness of integrating academic study with industrial experience.

1 Introduction

Embedded processing, where computers are used to monitor and control dedicated hardware, is a growing presence within mainstream computer science and engineering [1-3]. Network processing, where embedded processors monitor and control communication networks, is a premier example of embedded processing.

There is a growing need for undergraduate students to understand software-hardware co-design concepts and to acquire practical experience in embedded processing [4]. Network processing and network processor architecture provide an ideal context to teach software-hardware co-design at the advanced undergraduate level in computer science and engineering. In fact, network processor architecture is undergoing rapid evolution, making it a dynamic area for observation and contribution. The network systems in which network processors are deployed are also growing and evolving. These systems include substantial hardware and software components.

This paper describes an advanced undergraduate course that was designed and developed about network processing, integrated with a modular network processing laboratory, to bridge undergraduate learning and research in both software and hardware. Over the last several years there have been a number of graduate-level courses developed on network processors (e.g., [5]). We have adapted the graduate-level courses on network processors into an undergraduate-level course on network processing and processors. The course materials, including course notes and laboratory exercises have been developed and are freely available on the Internet to academic institutions teaching similar software-hardware co-design courses. A researcher from industry has co-taught the course, which adds valuable industrial experience in these fields to the course.

The modular network processing laboratory has been designed and utilized to teach undergraduate students in a “hands-on” manner the operation of a network processor as well as elements of network devices. There exist a number of papers that are useful references for designing lab sessions of this course. For example, [6-8] have discussed education of networking concepts via hands-on experiments or practical experience. We have observed that their course design can be improved by offering *safety-net* characteristics and industrial experience components. Safety-net means that students who fail to complete a particular assignment are still able to move forward to the next assignments and eventually get the incomplete part done. Experience with such a software-hardware combined environment will benefit students in the scientific, mathematical, and engineering disciplines.

2 Course Information

2.1 Components and schedule

The semester long *Network Systems Design* course consisted of four components: *Introduction*, *Traditional Network Systems*, *Network Processor Technology*, and *Example Network Processor*. They were divided into two categories: lectures and lab sessions. The course schedule is shown in Table I. The grade weights were assigned as follows: homework: 20%; midterm: 20%; lab projects: 30%; and final exam: 30%. There was no prerequisite on introductory computer network course and thus the first three weeks were used to introduce basic concepts of computer networks in a nutshell.

Table I. Lecture and Lab Schedule

Component	**Lecture**	**Lab**
Introduction (6 hours)	Course introduction, network architecture, layering & protocols, OSI and Internet architecture; Encapsulation, hardware building blocks, encoding, framing; Error detection, Ethernet (802.3), FDDI, switching and forwarding, circuit switching; Packet switching, IP, service model, socket, routing and forwarding; UDP and TCP.	Traffic monitoring and throughput measurement
Traditional Network Systems (12 hours)	Computer architecture; Packet processing algorithms and functions; Protocol software, socket; Hardware architecture for packet processing; Classification and forwarding; Switching fabrics.	Basic router configuration; Firewall, ethereal, switch vs hub
Network Processor Technology (6 hours)	Network processor introduction; Complexity of network processor design; Network processor architectures; Scaling a network processor; Design tradeoffs and consequences.	SystemC models and simulation
Example Network Processor (10 hours)	Overview of Agere network processor and FPL classification language; System architecture and modeling; Stateful network processor applications; Policing, buffer management and traffic shaping; Agere site visit; Network processing trends.	Network processor bridge; Fragmentation and Encapsulation; Stateful FPL application

2.2 Achieved Goals

The course offered in Fall 2003 consisted of both software and hardware components. The students were exposed to a variety of important software-hardware co-design concepts. They learned to program algorithms for network processing, use tools to design network processors, and construct network devices of complex network processing systems in a well-structured, hierarchical way.

We have created a project-based introduction to embedded processing in its application area of network processing, where there is increasing demand for skills and for which we anticipate substantial advances in technology. Students have gained hands-on experience in both the general area of embedded processing and in the specific area of network processing.

In addition, we have successfully strengthened ties between academic study of network processing and industrial practice in the field, given the fact that most advances in network processor architectures to date have been made in industry. Agere researchers have participated in the course development in terms of co-teaching lectures, developing laboratory sessions, conducting Agere site visit, and supervising internship.

Moreover, the development of the advanced undergraduate course in network processing has leveraged existing educational resources, including: (*i*) Classic texts and laboratory exercises in network processing before the advent of network processors, particularly Internet-oriented materials; and (*ii*) review feedback to *Network Systems Design Using Network Processors* (Agere Version of [9]), a new text by Professor Douglas Comer of Purdue University.

3 Network Processing Laboratory

3.1 Overview

The purpose of the network processing laboratory projects or assignments is for students to develop a thorough understanding of network processing concepts, architectures, algorithms and techniques by implementing them. "Learning through doing" forces the students to digest the information presented in classes to the point where they can instruct the computer how to apply it. Active learning such as this has a higher chance of having a lasting effect on students than if the students passively listen to lectures without reinforcement.

The architecture of the laboratory projects/assignments breaks the task of implementing network devices into smaller, more manageable chunks. They incrementally build on top of each other to incrementally create a complete hardware-software solution to a sophisticated network processing system.

A *safety net* was provided for students who fail to complete a particular assignment. We also offered the benefit in the laboratory that students have an opportunity to work with many different partners throughout the semester.

The overall approach of the laboratory sequence is to start with high-level, application-oriented networking concepts with which students are already familiar, such as Internet communications and the World Wide Web, and work our way down networking protocol layers in the examination of underlying protocols and their processing in software and dedicated hardware. Once we have explored underlying mechanisms, labs reverse their direction, examining how network processor architectures are evolving to handle higher-level protocol layers at full speed. Thus the laboratory sequence consists of an analysis stage leading to underlying mechanisms, followed by a synthesis stage that reveals the forces behind current trends in network processing evolution.

3.2 Lab Projects

Below is a list of the six incremental lab projects associated with the laboratory practice sessions.

(1) Traffic monitoring and throughput measurement (step 1 of analysis):

Initial exercises use ethereal/tcpdump and similar network traffic monitors to capture and observe live packets created by real applications such as web browsers and email. Concepts include generation and observation of structured traffic, a central activity in professional network processing. Students use traffic monitors and generators learned in this step in all subsequent steps.

(2) Basic router configuration and raw socket (step 2 of analysis):

The router configuration lab helps students to understand more of network protocols by configuring Cisco routers to support various network topologies of the local area network and architecture such as VLAN. Also a homework-oriented assignment of raw socket concentrates on conventional network programming interfaces used by protocols and applications.

(3) Firewall, ethereal, switch vs. hub (step 3 of analysis):

Projects place network interface cards (NICs) on conventional computers into promiscuous mode and control packet receipt and transmission directly. This lab session includes three parts: configuring firewalls using `iptables` in Linux; using `ethereal` to capture network packets and observe the packets in various layers; and comparing the difference between a switch and a hub.

(4) SPA network processor simulator (step 1 of synthesis):

At this stage we begin reworking the mechanisms used in the previous stages into a form supported by dedicated network processing instruction sets and multiprocessor topologies. Exercises begin with an examination of fast path (a.k.a. wire speed or hard real-time) processing as contrasted with slow path (a.k.a. control path or non-real-time) processing, using both high-level functional simulation environments and actual network processor development environments, including tools from Agere Systems, Inc.

(5) SystemC models and simulation (step 2 of synthesis):

This stage uses high-level, functional simulation to explore hardware building blocks such as pattern matchers that are part of network processors. Students complete design of a hardware block and simulate its interactions with other network processor components. In a complete system design, a system designer can simulate execution of network processing code such as routing on a simulated processor written in SystemC. Students exercise this two-tiered simulation of hardware and software called co-simulation.

(6) Stateful FPL application (step 3 of synthesis):

Exercises focus on a representative sample of programs illustrating how network processors are currently used. Examples include bridges, routers, network address translators, and firewalls. For example, using Agere FPL (Functional Programming Language) to deploy a hash table to implement a learning Ethernet bridge.

4 Student Response

In a survey question asking students' comments about the course, 40% of the students mentioned that they liked the hands-on labs, and 10% of these students stated that the later labs tied everything together. In addition, 40% students found the subject matter to be relevant to today's network field. They felt that the material was interesting and presented well, and they learned a lot of new material. Other students appreciated that the professors were well-qualified and treated the students with respect.

Twenty percent of the students felt that the course could be split over two semesters, with the first semester introducing the basics of network system design and the second semester introducing more advanced topics in greater detail. Other suggestions by individual students included having more labs like the first two, and providing more real work and fewer simulations.

5 Conclusions and Future Work

This paper presents contents of the *Network Systems Design* course used to introduce undergraduate students to understanding software-hardware co-design concepts and acquiring practical experience in embedded processing. The goals achieved include: (*i*) carrying out lab-based introduction to embedded processing in its application area of network processing, and (*ii*) strengthening ties between academic study of network processing and industrial practice in the field.

The next time this class is taught, a prerequisite of an introductory undergraduate course on computer networks should be imposed and the number of lectures and labs on the introduction of networking concepts would probably be increased. In addition, the student presentations on "what I learned" would be reserved for the second half of the semester.

More Agere's software will be adapted to our undergraduate course. Currently there is a production-quality network processor simulator (System Performance Analyzer – SPA) that Agere has donated for use in the course. There are also two prototype software tools that teaching assistants could enhance for use in the course. One is a network processor emulator (SAUNA) that translates network processor code into C code that can run on a PC containing two network interface cards. This emulator will allow students to design and test network processor algorithms on inexpensive PC hardware; code runs at PC speeds rather than at faster network processor speeds, but algorithms work identically. Having the emulator in addition to actual network processor hardware supports more lab stations at low expense, and it scales readily to inexpensive reuse at other colleges and universities. The other prototype software tool is an open source embedded system debugger from Agere (RTEEM) that a teaching assistant will enhance for debugging and algorithm visualization of network processing programs running on the emulator.

Acknowledgment

This project has been partly financed by a grant from the Commonwealth of Pennsylvania, Department of Community and Economic Development, through the Pennsylvania Infrastructure Technology Alliance (PITA), and it has also been supported by National Science Foundation DUE CCLI Award #0310745 and donations from Agere Systems, Inc.

References

[1] W. Bux, W.E. Denzel, T. Engbersen, A. Herkersdorf, and R.P. Luijten, "Technologies and building blocks for fast packet forwarding," *IEEE Communications Magazine*, Vol. 39, No. 1, Jan. 2001, pp. 70–77.

[2] T. Wolf and J.S. Turner, "Design issues for high-performance active routers," *IEEE Journal on Selected Areas in Communications*, Vol. 19, No. 3, March 2001, pp. 404–409.

[3] Y. Coady, S. O. Joon, and M.J. Feeley, "Using embedded network processors to implement global memory management in a workstation cluster," *Proceedings of the Eighth International Symposium on High Performance Distributed Computing*, 1999, pp. 319–328.

[4] P. Paulin, F. Karim and P. Bromley, "Network processors: a perspective on market requirements, processor architectures and embedded S/W tools," *Proceedings of the DATE* 2001 *on Design, Automation and Test in Europe*, 2001, pp. 420–429.

[5] Network processor homepage at Purdue University, http://www.cs.purdue.edu/np/.

[6] M. McDonald, J. Rickman, G. McDonald, P. Heeler, and D. Hawley, "Practical experiences for undergraduate computer networking students," *Journal of Computing in Small Colleges*, Volume 16, Issue 3, March 2001.

[7] K. M. Sivalingam and V. Rajaravivarma, "Education of wireless and ATM networking concepts using hands-on laboratory experience," *ACM SIGCSE Bulletin*, *Proceedings of the Thirtieth SIGCSE Technical Symposium on Computer Science Education*, Volume 31, Issue 1, March 1999.

[8] P. Steenkiste, "Networks: a network project course based on network processors," *Proceedings of the* 34*th Technical Symposium on Computer Science Education*, February 2003.

[9] D. Comer, *Network Systems Design Using Network Processor*, Prentice Hall, Upper Saddle River, New Jersey, USA, 2003.