Teaching Digital Design to Computing Science Students in a Single Academic Term

José Nelson Amaral,

Paul Berube, Paras Mehta

Department of Computing Science

University of Alberta, Edmonton, Canada

{amaral, berube, paras}@cs.ualberta.ca

Mailing Address:

José Nelson Amaral

Department of Computing Science

University of Alberta

Edmonton, Alberta

Canada, T6G 2E8

Phone: (780) 492-5411

Fax: (780) 492-1071

Abstract

How should digital design be taught to Computing Science students in a single one-semester course? This paper advocates the use of state of the art design tools and programmable devices and presents a series of laboratory exercises to help students learn digital logic. Each exercise introduces new concepts and produces the complete design of a stand-alone apparatus that is fun and interesting to use. These exercises lead to the most challenging capstone designs for a single semester course of which the authors are aware. Fast progress is made possible by providing students with pre-designed input/output modules. Student feedback demonstrates that the students approve this methodology. An extensive set of slides, support teaching material, and lab exercises are freely available for downloading.

Keywords: Digital logic design, digital systems, teaching laboratory, Field-Programmable Gate Arrays (FPGA).

1 Introduction

The teaching of digital design in Electrical and Computer Engineering (ECE) curricula is well established. However, the teaching of digital design to Computing Science (CS) students has not been discussed at length in the engineering education literature. Additional constraints in a CS program include (1) reduced number of hours dedicated to hardware-related subjects; and (2) incoming students' lack of background on switching theory, analog circuits, electronics, and limited exposure to the concepts of concurrency, feedback, and timing.

Even when incoming CS students lack what would be considered essential prerequisites in an ECE program, a well planned, one-semester course on digital design can produce adequate state-of-the-art training on digital design, expose students to key digital design principles, and allow the students to design and implement non-trivial apparatuses that work. This paper presents a methodology for the teaching of digital design in a single semester course in a Computing Science program.

Section 2 examines alternative approaches to teaching logic design. Lecture material and teaching methodology are discussed in Section 3. Then Sections 4, 5, and 6 present the laboratory environment and exercises, and Section 7 presents results from the student evaluation of the class.

2 Related Approaches

This section reviews experiences and reflections about the transition from plug-boards to programmable logic.

The digital logic education literature provides extensive arguments *against* starting design labs

with complex devices such as FPGAs. Kleinfelder *et al*., Areibi, and Nickels advocate that when transitioning from low density logic devices mounted in plug-boards to programmable logic, digital design classes should retain some component of non-programmable small or medium-scale integrated circuits [1, 2, 3]. Newman *et al*. chose to make the transition from plug-boards to programmable logic devices (PLDs), which are simpler than FPGAs [4]. Nixon argues that the complexity of FPGAs are inappropriate for a first course on logic design [5]. Most of these observations stem from experiences within an ECE program where more time (in comparison with a CS program) is dedicated to digital design.

In order to reach the state of the art in one term, the authors forgo the "touch and feel" experience awarded by lower scale integration. With careful planning and support material, an aggressive schedule of increasingly complex designs can be successfully implemented in one semester. Although the often extolled debugging of wire connections is absent from this lab environment, the use of an external keyboard, audio-set, Light Emitting Diode (LED) displays and pushbuttons gives the students an appropriate level of interaction with the devices that they design.

Once FPGAs are selected to be used from the start of the course, a set of design tools must be chosen. Calazans and Moraes suggest that the availability of design tools and FPGAs allows the combined teaching of computer architecture and digital design early in a CS program (third term), and they recommend a combination of industry-grade and educational tools [6]. Although some authors of educational tools are their strongest advocates (such as Rodríguez-Pardo *et al*. [7]), others argue for industry-grade tools [8]. In a curriculum with a single digital design course, the use of industry-grade tools is recommended because they afford the students the best training for

a career that might include logic design.

Instructors interested in switching to FPGA-based digital design laboratories will also find relevant description of experiences in [9]-[10]. The course plan described here is distinct from these experiences because it focus on transitioning from very basic design skills to challenging capstone designs in one term. This goal is accomplished through the use of supporting modules for I/O operations.

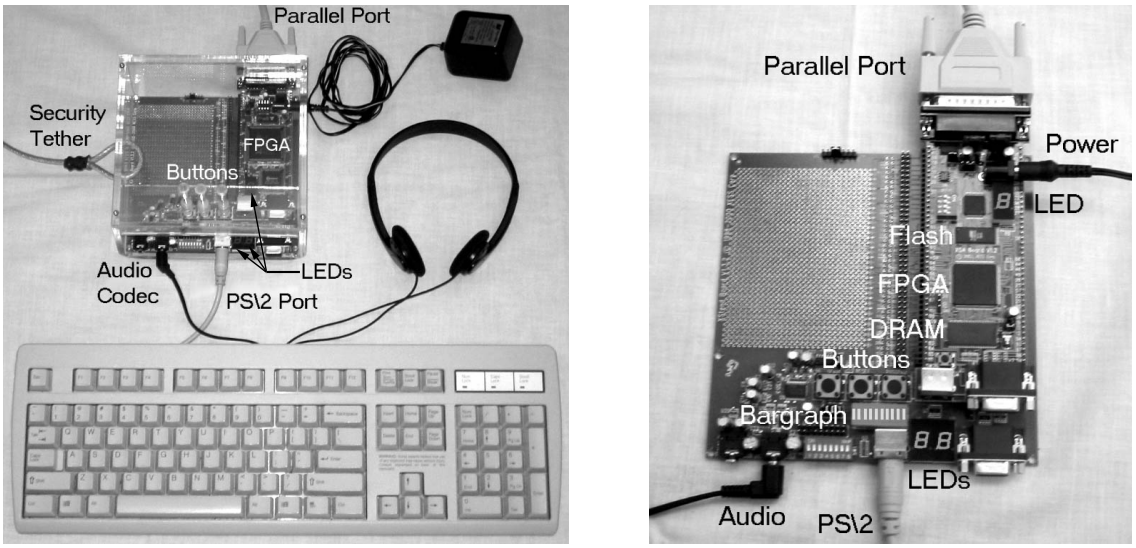
3 Lecturing Methodology

In-class time is effectively used by relying on an extensive set of slides for the lectures [11]. These slides are carefully designed to allow in-class interaction with the students.[1] Animation techniques are often used to allow in-class quizzes. Students are asked to put printout of slides aside and to take out paper and pencil to solve these quizzes. Logic gate construction follows Yale Patt's abstraction of light switch to represent a transistor [12]. The initial chapters of the adopted textbook, by John F. Wakerly, discuss number systems and electronic technology [13]. This material is made into reading assignments with scheduled 10-minute in-class quizzes. Homework assignments are regularly assigned, but in order to rationalize Teaching Assistants' (TAs) time the solutions are not collected for grading. Instead, short, scheduled, in-class quizzes test the same material at the homework due-date.

The use of FPGAs requires the inclusion of VHSIC Hardware Description Language (VHDL)[2] learning in the one-term course. The lab exercises were designed to minimize the in-class time

[1] Slides, exams, quizzes and lab assignments are publicly available at *www.cs.ualberta.ca/∼amaral/courses/329*.

[2] VHSIC stands for Very High Speed Integrated Circuit.



(a) XSA-50+XStend, in custom enclosure

(b) XSA-50+XStend

Figure 1: Development boards as used in the laboratory, from XESS Corp

dedicated to VHDL. The students are told on the first day of class that they are expected to learn VHDL on their own.[3] Once this level of expectation is set, only two 50-minute lectures are required to discuss the most important principles of VHDL modeling in class.

The careful use of technology and old fashioned common-sense allows the development of a challenging but exciting one-term digital design course for a CS curriculum. Students now look forward to, and praise (Section 7), this third year optional class in the program.

4 Laboratory Environment

The exercises where used in a laboratory with 20 workstations, each workstation with a XSA-50 FPGA board and an XStend daughter-board.[4] Desk fastened custom plastic enclosures minimize wear and tear, accidental damage, or unsupervised removal and allows 24-hour operation for the lab. Figure 1(a) shows the enclosed boards connected to a parallel port of a host computer, to a power supply, to a PS\2 keyboard, and to a headphone set. Figure 1(b) is a closer view of the XStend and the XSA-50 outside the plexiglass encasing.

4.1 Hardware and Software Environments

The XStend boards have an 8-segment bar-graph LED used to display binary values, a 7-segment digital LED used as an alphanumerical display, and a stereo codec used for audio output. Inputs are accepted through the buttons on the XStend board, through the parallel port data lines, and from a keyboard attached to the PS\2 port. Students are provided with a module to interface with the keyboard. Once learned, this interface is re-used in several exercises. The XSA's 8MB Dynamic Random Access Memory (DRAM) and a 128KB Flash memories are used to create designs that function without constant input from the user.

The Xilinx Integrated Software Environment (ISE) 5.2i is used for design entry. ISE presents a hierarchical view of the design process which clearly illustrates the dependencies and sequencing of implementation tasks. VHDL Simili from Sonata Electronic Design Automation (EDA)

[3] In this case the CS background is an advantage because third year CS students are familiar with learning new programming languages on their own.

[4] XSA-50 and XStend are products of the X Engineering Software Systems (XESS) Corporation (www.xess.com).

is used for testbench-based simulation. VHDL Simili is more intuitive and user friendly for beginner students than the industry-strong ModelSim from Mentor Graphics. Testbench writing is challenging to the novice designer, thus students are provided rudimentary testbenches for all but the capstone labs.

4.1.1 Testbenches

When using a testbench the student can detect errors and identify signals that are assigned incorrectly. Testbenching also allows for easier grading of lab exercises. Teaching assistants (TAs) are provided with testbenches that are not published to the students, and hence can do a thorough, fast and fair comparison of designs. A simplified testbench driver is supplied to allow students to perform initial testing, and to provide the basis for a more complete testbench. Testbench inputs for self-checking testbenches are supplied as a text file, and the corresponding correct outputs are provided in another file. If simulated outputs do not match supplied outputs, an error message is generated.

4.1.2 Demonstration

Successfully simulated designs may not necessarily work in hardware since timing issues and hazards may not be simulated completely. Also, complex behaviors, such as the PS\2 protocol, are not simulated; instead, the PS\2 output is simulated. Hence, if the keyboard interface is not correctly connected to the keyboard pins, this error would not be simulated. Conversely, unsuccessful simulations do not imply total failure. To allow partial marks for partial functionality, demonstration is the only way to assess the extent of the student effort.

5 Term Lab Exercises

This section provides only brief descriptions of the first five labs. Complete descriptions and VHDL modules are found in the course web-page. Initial lab exercises familiarize the students with the integrated development environment and with the design of combinatorial circuits.

Simple Alarm System. Design the logic for the control circuit of an alarm from a natural language specification. The students are required to build both a schematic diagram and a VHDL design so that they can contrast the advantages of both input formats.

Parity Checker. Design a circuit to read a hexadecimal digit from a keyboard and calculate its parity. The parity is displayed as a numerical digit on a 7-segment LED, while the input is represented in binary on a bar-graph LED. This lab introduces the board's input/output features, structural VHDL, and the VHDL Simili simulator.

Treehouse Encryption. Implement a bit-scrambling encryption algorithm. A standard PS\2 keyboard is used as input. Complete modules in VHDL and partially completed VHDL code are provided for the students to examine. Portions of the design are provided as specifications that require the students to build a hierarchical design using submodules.

Scrolling Message Display. Produce a scrolling message in the two neighboring 7-segment LEDs of the XStend boards. The message to be displayed is stored in an `Off-Chip Memory` in ASCII format. The end of the message is detected by a special ASCII code. When the end of the message is reached, the message has to be scrolled in reverse order until the beginning of the message is reached, whereupon it will scroll forward again. A PS\2 keyboard

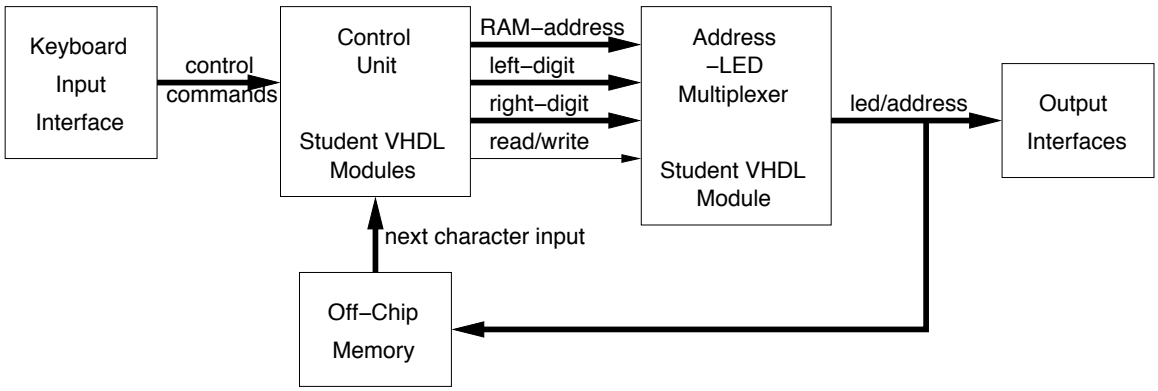


Figure 2: Block diagram for the *Scrolling Message Display* system.

is used to control the speed of the scrolling. A block diagram for this lab is shown in Fig. 2.

Besides the VHDL code for the input and output interfaces, the students are also given a skeleton for the design of the `Address-LED Multiplexer` and for the `Control Unit`. Completion of the code in these skeletons will produce the basic functionality of the simple scrolling device. However, to obtain the continuing forward-reverse scrolling function, the `Control Unit` must be re-designed. With a functional forward scrolling system, the use of a hierarchical design for the new `Control Unit` should be natural. Also, the control of the scrolling speed through the keyboard interface requires that the students implement a variable counter in the `Control Unit` to determine how often a new character has to be read from the Off-Chip Memory. This exercise introduces counters, multiplexing, and component reuse, and practices hierarchical design.

Multi-Mode Calculator. Design a finite state machine that implements a two-digit hexadecimal calculator that handles input expressions in *prefix*, *infix*, or *postfix* form. The PS\2 interface is used both to select the operation mode, and to input the operations to be performed by

the calculator. The 7-segment LEDs display the result of the calculations and the current operation mode. Numerical inputs are two digits. Addition, subtraction, and multiplication operations are supported in all modes.

This calculator can be implemented as a single large finite state machine (FSM) in which numerical inputs, arithmetic operators, and commands to change mode of operation are the inputs. However, a better strategy is for the student to break this FSM into smaller ones. For this lab no block diagrams are provided, and the students are free to build their design however they desire.

6 Capstone Designs

The final lab exercise is the capstone for the class. A collection of different exercises can be rotated as the capstone design in order to keep the course interesting and introduce some variation in the undergraduate program. Currently there are two capstone designs: a music recorder and an interactive game.

6.1 Music Recorder

The *Music Recorder* is the design of a musical keyboard played through the audio codec whose output can be recorded to and played back from the RAM. The design must also be able to transpose the music up or down by as much as one octave. The amount of transposition is set from the keyboard. Students are given only a natural (*i.e.*, English) language specification of the design.

Students are provided with the keyboard interface and a series of VHDL modules that produce an audio interface with the codec that translates Musical Instrument Digital Interface (MIDI) note signals into sounds. Students are responsible for the design of the `Control Modules`. As a bonus, the students may expand their design to display the transposition of notes on the LEDs. This requirement demands that shared lines between the LEDs and the RAM memory be multiplexed.

Important features of this lab include: (a) tones must be held while a keyboard key is depressed and stopped when the key is released (therefore, both key presses and releases must be monitored); (b) RAM must be accessed for both reading and writing; meaning control signals must be properly generated; and (c) control inputs can arrive from the keyboard asynchronously. A successful design depends on the proper interaction of several fairly complex state machines.

6.2 Type, Type, Revolution!

Type, Type, Revolution! is based on a popular arcade game of similar name, "Dance, Dance, Revolution!" This assignment involves designing an interactive game using FPGAs. Students are given only high-level design specifications for this game.

The game itself is fairly simple: the system reads an instruction from memory, selected from a set of directional keys (up, down, left, right), and displays the instruction on two 7-segment LEDs on the XStend board. The user must then press the key corresponding to the displayed instruction before the next instruction is displayed. Three mistakes or missed keys are allowed before the game ends. At the end of the game the number of sequences completed is displayed. When the sequence ends, the user is credited one additional allowable mistake (to a maximum total of

three), and the sequence repeats. Students are also encouraged to develop additional functionality for bonus marks. Suggestions include increasing the game speed after each completed sequence and making the length of time for each key variable.

7 Student Feedback

This section presents results from a standard student evaluation of courses and a summary of a qualitative evaluation of the labs through written comments in the lab reports.

Year	# of Students Registered	# of Responses
2000	30	20
2001	37	31
2002	46	31
2003	43	38

Table 1: Number of respondents to the student evaluation of the course.

At the University of Alberta, standard student evaluations of courses are conducted in every term. Table 1 shows the number of students registered and the number of responses to the in-class student evaluation for this course.

Table 2 reports the answers to four questions that should be relevant for instructors interested in adopting a similar teaching methodology for digital design. The class was taught by the same instructor in all four years. The numbers under the "Percentile" column indicate how the answers compare with other courses taught during the same term at the University of Alberta. For instance $> 75\%$ of 162 means that when compared with a cohort of 162 courses that offer laboratory, the answers for 2000 were more positive than 75% of the courses.[5] For the other three questions,

[5]The size of the cohort for some questions and years was not published by the university, but it should be similar

Year	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	Percentile
The lab environment was appropriate for this course						
2000	0	1	4	9	6	$>$ 75% of 162
2001	6	3	9	12	1	$>$ 25% of 274
2002	2	2	9	13	5	$>$ 50% of 385
2003	0	2	8	16	12	$>$ 75% of 483
In-class time was used effectively						
2000	1	1	2	10	6	$\geq$ 50% of 8510
2001	0	1	3	20	7	$\geq$ 50%
2002	0	1	0	19	11	$>$ 50%
2003	0	0	1	10	27	$>$ 75%
The workload for this course was appropriate						
2000	1	2	9	7	1	$<$ 25% of 9840
2001	5	8	8	9	1	$<$ 25%
2002	3	5	11	10	2	$<$ 25%
2003	2	3	7	16	10	$>$ 25%
Overall, the quality of the course content was excellent						
2000	1	1	3	7	8	$>$ 50% of 8523
2001	0	0	7	19	5	$\geq$ 50%
2002	1	1	6	15	8	$\geq$ 50%
2003	0	0	4	15	19	$>$ 75%

Table 2: Student's reaction to various statements evaluating the course

the answers are compared with all sessions evaluated in that term at the University of Alberta. The disruption caused by the adoption of VHDL and FPGAs caused a significant reduction in the number of students who consider the lab environment appropriate in 2001 and 2002. However, on the third edition of the new course (2003), the student responses were more positive than before these changes.

As glitches were eliminated from the lab experiments, the software environment and lecture material were improved. The instructor acquired experience teaching the class, thus the evaluation of the course improved significantly. The most significant changes are in the number of students

with previous years.

who consider the workload appropriate and the course content excellent. The workload of this course is higher than in similar courses in other universities, and it is also higher than similar courses in the same term at the University of Alberta. However, over the three-year period in which the changes were implemented, the expectation of the students changed, and the additional effort required for the course is now considered "normal" by the students. A clear evidence of the students' approval of the changes and of their benefit from the course is that this elective class and the enrollment has not dropped in spite of the additional effort that the class requires.

Review of Student Comments

Students are requested to include in the report submitted with each lab a brief evaluation of the lab exercise. In this evaluation they discuss the lab specification and indicate any problems that they had with the lab presentation. During the annual updating of the labs, these comments are reviewed to make corrections for the next year. Here is a summary of the comments about the capstone labs (Music Recorder in 2001; Type, Type, Revolution! in 2002 and 2003):

- In 2001, students were given less guidance in early labs; thus they were exposed to the process of conceptualizing a design from a natural language specification earlier. They found Lab 6 challenging but not overburdening.

- In 2001, students frequently wrote that they were excited to have designed and implemented an apparatus "that does something" (play music).

- In 2002, block diagrams and conceptual designs to the earlier labs were added in an attempt to help out the students. This addition proved to be a wrong decision because the students

felt that there was a major gap between the earlier labs and Lab 7.

- Students felt that timing issues were not discussed in class to the extent needed in the lab. This class is offered to CS students, and they have a hard time working with difficult to reproduce behavior during testing. The problem was corrected in 2003 by including one extra lecture on timing and by warning students at the start of the term that they could encounter non-deterministic behavior in the lab.

In order to address these problems less design information (no block diagrams) was provided in earlier labs in 2003, and the discussion of design flow and timing issues in the classroom was expanded. Also, test benches were introduced to allow more comprehensive testing during simulation, and the two first lab exercises were redesigned to emphasize VHDL-based, as opposed to schematic, design input.

Although sometimes students spend frustrating hours in the lab because of the challenging nature of the exercises, student comments are very positive and indicate an enjoyment of the exercises. Students were included in the process of improving the labs, and they often offered constructive feedback. While reading the comments one cannot fail to feel the sense of accomplishment and triumph over a challenge that many students expressed.

8 Final Remarks

This paper addresses the problem of introducing Computing Science students to state-of-the-art digital design. This task often falls to ECE faculty or to CS faculty with ECE background. The experience described here demonstrates that with careful planning and well developed lecture

material and lab exercises, CS students can be offered an engaging and rewarding digital design experience. The material presented in this paper is publicly available in the course webpage in the hope that it will be of benefit to colleagues elsewhere.

Acknowledgements

Students were supported by Summer Fellowships from the Natural Sciences and Engineering Research Council (NSERC) of Canada. Xilinx Co. donated the boards and software for the lab.

References

[1] S. Areibi, "A first course in digital design using VHDL and programmable logic," in *31st ASEE/IEEE Frontiers in Education Conference*, vol. 31. ASEE/IEEE, October 2001, pp. 19–23.

[2] M. W. Kleinfelder, M. D. Gray, and L.-C. G. Dudevoir, "A hierarchical approach to digital design using computer-aided design and hardware description languages," in *29th ASEE/IEEE Frontiers in Education Conference*. ASEE/IEEE, November 1999, pp. 18–22.

[3] K. Nickels, "Pros and cons of replacing discrete logic with programmable logic in introductory digital logic courses," in *Proceedings – 2000 ASEE Annual Conference*. ASEE, 2000.

[4] K. E. Newman, J. O. Hamblen, and T. S. Hall, "An introductory digital design course using a low-cost autonomous robot," *IEEE Transactions on Education*, vol. 45, no. 3, pp. 289–296, August 2002.

[5] M. S. Nixon, "On a programmable approach to introducing digital design," *IEEE Transactions on Education*, vol. 40, no. 3, pp. 195–206, August 1997.

[6] N. L. V. Calazans and F. G. Moraes, "Integrating the teaching of computer organization and architecture with digital hardware design early in undergraduate courses," *IEEE Transactions on Education*, vol. 44, no. 2, pp. 109–119, May 2001.

[7] L. Rodríguez-Pardo, M. Moure, M. Valdés, and E. Mandado, "VISCP: A virtual instrumentation and CAD tool for electronic engineering learning," in *1998 Frontiers in Education Conference*. ASEE/IEEE, 1998.

[8] G. Puvvada and M. A. Breuer, "Teaching computer hardware design using commercial CAD tools," *IEEE Transactions on Education*, vol. 36, no. 1, pp. 158–162, February 1993.

[9] E. I. Boemo, "Computer-based tools for electrical engineering education: Some informal notes," in *Proceedings of Computer Aided Engineering Conference 1999*, 1999, pp. 7–13.

[10] A. Leva, "A hands-on experimental laboratory for undergraduate courses in automatic control," *IEEE Transactions on Education*, vol. 46, no. 2, pp. 263–272, May 2003.

[11] J. N. Amaral, "Cmput329 webpage," http://www.cs.ualberta.ca/ amaral/courses/329, 2003.

[12] Y. N. Patt and S. J. Patel, *Introduction to Computing Systems: from bits & gates to C & beyond*. McGrawHill Press, 2001.

[13] J. F. Wakerly, *Digital Design Principles & Practices*. Prentice Hall, 2002.

Biographies

José Nelson Amaral received the Ph.D. in Electrical and Computer Engineering from the University of Texas at Austin, in 1994, the M.E. from the Instituto Tecnológico de Aeronáutica, São José dos Campos, SP, Brazil, in 1989 and the B.E. from the Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS), RS, Brazil, in 1987. He is a Professor in the Department of Computing Science at the University of Alberta, Canada. He was a post-doctoral researcher at the University of Delaware from 1998 to 2000, and was Associate Professor of Electrical Engineering at PUCRS before that. His previous research includes theory and applications of Artificial Neural Networks, Combinatorial Optimization Problems, Parallel Architectures for Symbolic Processing, and Multi-Threaded Architectures and Programming Models. His current research interests include Compiler Design and Optimization, Cache-Conscious Algorithms, Applications of Programmable Logic, and High-Performance Computer Systems. Dr. Amaral is a Senior Member of the IEEE, an associate editor for the *IEEE Transactions on Computers*, and served in the organization of many international conferences.

Paul Berube received a B.Sc. in Computing Science from the University of Alberta in 2003. He was awarded a competitive Natural Sciences and Engineering Research Council (NSERC)

of Canada graduate fellowship in 2003 and is currently pursuing a M.Sc. in Computing Science at the University of Alberta. His research interests include Compiler Design and Optimization, Computer Architecture, and Programmable Logic Devices.

Paras Mehta received the B.Sc. in Computing Science from the University of Alberta in 2004. He received a Undergraduate Student Research Awards (USRA) from NSERC in 2003 and 2004. Mr. Mehta is a recipient of the Terence Holowach Memorial Prize and of the Amdahl Academic Achievement Scholarship in Computing Science. His currently research interest is on parallel programming systems.