AC 2009-816: INTRODUCING RECONFIGURABLE COMPUTING IN THE UNDERGRADUATE COMPUTER ENGINEERING CURRICULUM

Arun Ravindran, University of North Carolina, Charlotte

Patricia Tolley, University of North Carolina, Charlotte

Arindam Mukherjee, University of North Carolina, Charlotte

Introducing Reconfigurable Computing in the Undergraduate Computer Engineering Curriculum

Abstract

We present our curriculum development efforts on introducing undergraduate computer engineering seniors to the emerging paradigm of high performance computing through the use of FPGA based reconfigurable computers. The prerequisites required for the course are programming using a high level language such as C/C++ or Java and an understanding of logic design, both which a typical undergraduate computer engineering student acquires at the sophomore or the junior level. An associated laboratory component was also developed, where weekly hands-on laboratory sessions serve to reinforce the ideas learned in the lecture. The course projects are drawn from a variety of disciplines which use high performance computing including bioinformatics, scientific computing, and signal processing. The course was assessed through pre and post tests, focus groups, and external evaluators drawn from faculty from other departments. Our assessments indicate that the course has had a significant impact on student understanding of digital logic design, exploitation of data parallelism in computationally intensive algorithms, and hardware-software integration issues. Our overall conclusion is that with a carefully planned syllabus, course projects, and the availability of student support resources, introducing reconfigurable computing to undergraduate computer engineering students can be a useful vehicle for teaching topics on parallel hardware and parallel algorithms.

Introduction

The availability of high speed Field Programmable Gate Arrays (FPGA) with more than a billion transistors has provided hardware designers with a platform for implementing complex high performance designs such that the programmability of general purpose processors and the performance of custom-designed hardware can be simultaneously achieved[1]. In a reconfigurable computer, FPGAs in conjunction with microprocessors serve as hardware accelerators for acceleration of computationally intensive tasks, delivering significant increase in performance[2]. However, the typical undergraduate computer engineering curriculum has yet to introduce students to this rapidly emerging computing paradigm.

Over the past couple of years, the authors, with the help of a National Science Foundation Course, Curriculum and Laboratory Improvement Grant, have developed and taught a new course titled “Hardware Acceleration using Field Programmable Gate Arrays (FPGAs)”. The course seeks to introduce undergraduate computer engineering seniors to the emerging paradigm of high performance computing through the use of FPGA based reconfigurable computers. An associated laboratory component was also developed, where weekly hands-on laboratory sessions serve to reinforce the ideas learned in the lecture. This paper focuses on three key components that describe our approach in educating computer engineers in this area of contemporary interest.

1) Emphasis on the use of systolic array architectures for efficient hardware design on FPGAs and the tight integration of laboratory and lectures.

2) Use of multi-disciplinary projects drawn from diverse areas such as scientific computing, signal processing, and bioinformatics that expose students to the use of reconfigurable computing in real life high performance computing applications.

3) Continuous assessments using pre-test, post-tests, midterm examinations, and in-class focus groups. We note that given the broad scope and challenging nature of the material covered in a semester, continuous assessment is necessary to maintain good communication between the instructor and the students and to achieve specified learning outcomes.

Curriculum fit

At our university, computer engineering undergraduate students take the following course sequence prior to our new course:

Logic System Design: One semester sophomore course offered every semester where students are introduced to Boolean logic and combinational logic design.

Digital Design Laboratory: One semester sophomore course offered every semester involving laboratory implementation of combinational logic circuits.

Advanced Digital Logic Systems: One semester junior course offered every semester (including summer) which includes sequential logic design, data path components, register-transfer level design, introduction to the hardware description language VHDL, and a laboratory component involving use of FPGAs for logic prototyping. Advanced Digital Logical Systems is the pre-requisite for the new course. Typical enrollment is around 40 students in the fall semester and 30 students in the spring semester.

A survey of computer engineering curricula in other universities indicates that computer engineering undergraduates have used FPGAs for logic prototyping early in the junior year making it feasible to introduce this course in the first semester of their senior year. The new course could serve as a depth requirement in some programs, either as a required course or as an elective depending on the programs goals.

At our institution, the new course is a required depth course taught both in the Fall and the Spring semesters.

Course Components

Course objectives

At the end of the course the students must have the following knowledge and skills.

(1) Ability to model and synthesize digital designs in VHDL.

(2) Familiarity with FPGA architectures, design flow, and interfacing issues for use in high performance computing.

(3) Exposure to diverse computationally intensive algorithms from digital signal processing, scientific computing, and bioinformatics

(4) Use of systolic array architectures for exploiting fine grained data parallelism on FPGAs.

Digital design using VHDL

The computational kernels to be implemented on the FPGA need to be expressed in a hardware description language or a C dialect that can be compiled to run on the hardware[5]. Since our students have had prior exposure to VHDL, we use it to capture and synthesize logic design on the FPGA. Also, VHDL provides better control and flexibility over the final design compared to high level C dialects such as ImpluseC[6] and MitrionC[7]. Although the students taking this course have had significant exposure to digital design and some exposure to VHDL, we spend about one week reviewing topics such as sequential logic and register transfer logic design. A further three weeks is dedicated to an in-depth teaching of VHDL with topics including VHDL structure and syntax, behavioral, structural and data flow modeling, writing test benches, and development of synthesizable register-transfer level code. The lecture slides are made available online on the course webpage [3] enabling students to preview the material before the class. The book by Ashenden [4] is used as a required text book.

FPGA Design Flow

The students are introduced to the hardware architectures of high performance FPGAs such as the Xilinx Virtex family of FPGAs [8]. Special emphasis is placed on advanced architectural features such as Block RAMS (BRAMs), serial transceivers, embedded multipliers, and multiple Power-PC hardcore processors. The students are introduced to the FPGA design flow consisting of design capture in VHDL, simulation for functional and timing verification, synthesis, place and route, and post place and route simulation. In a typical undergraduate FPGA laboratory with an emphasis on logic prototyping, the FPGA is interfaced to the external processor through a relatively low speed parallel port. However, to achieve the high speed data transfer required for hardware accelerator applications, the students are made familiar with direct interfacing of FPGA boards with the host processor through the PCI bus and the use of specialized interconnects by reconfigurable computer vendors such as SGI Inc. and Cray Inc. Also, the

students are exposed to the concept of building a scalable system using multiple FPGA boards with gigabit transceivers. Emphasis is placed on data transfer limitations, and the students are made aware of the importance of analyzing both computing and memory latencies when using FPGAs as computational platforms. We spend about one week on this topic.

Systolic array computing

The low clock speed of FPGAs compared to microprocessors makes it necessary for the hardware designer to take advantage of as much parallelism as possible. We introduce the students to the concept of systolic array architecture as a means of achieving good computational efficiency on FPGAs. Systolic arrays are regular arrays of simple finite state machines where each finite state in the machine is identical and connected locally in a regular fashion. Systolic array architectures have the critical advantages of modularity, regularity, local interconnection, and highly pipelined multiprocessing. Students are made aware of the memory, bandwidth and routing limitations on an FPGA and how the properties of systolic arrays lead to scalable designs on FPGAs.

Systolic arrays operate best on systolic algorithms where the data item is not only used when it is input but is also reused as it moves through the pipelines in the systolic arrays[9]. Systolic algorithms can be found in many computationally intensive applications such as digital signal processing, scientific computing, and bioinformatics. Though a formal formulation and analysis of systolic algorithms is out of the scope of this course, students are introduced to the design of systolic algorithms through familiar examples such as discrete time convolution and dense matrix multiplication. The students are first provided with a sequential code fragment in C implementing the algorithm. Small test cases are then used to demonstrate the formulation of the systolic algorithm from the sequential code. For example, in a 1D discrete time convolution algorithm, for the case of an 8 long input discrete signal {x[1]…x[8]} and an impulse response {h[1] …h[4]}, the students are shown that the output sequence {y[1]…y[11]} can be realized using the systolic array shown in Figure 1.

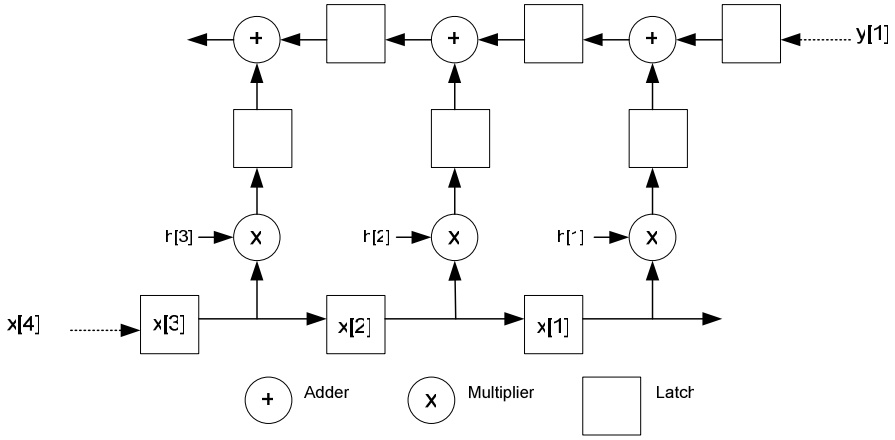

Figure 1. Systolic array for 1D discrete time convolution.

We spend approximately two weeks on this topic.

Laboratory assignments

A significant feature of the new course is the tight integration of the lecture with the laboratory. The course meets for one hour and 15 minutes twice a week, once in the classroom for the lecture and then in the laboratory for programming assignments. We have developed a series of laboratory exercises that serve to reinforce the concepts introduced in the class and the familiarize students with simulation and synthesis tools. A short pre-laboratory assignment is given to ensure that the students are adequately prepared for the laboratory session. The instructors ensure their availability in the lab to answer questions that students have while doing their lab assignments. Additionally a graduate teaching assistant is available during the lab session as well as for a few hours outside class each week to answer student queries. Our lab sessions are currently based on VHDL and the Xilinx ISE which includes the ModelSim simulator and the Xilinx XST synthesizer. The hardware platform consists of the PCI-X based Nallatech FPGA boards equipped with Virtex-II 6000 FPGAs and 128 MB on-board SRAMs.

Course Projects

The design project forms a crucial part of the new course by exposing students to real world computationally intensive algorithms drawn from diverse areas such as signal processing, scientific computing, and bioinformatics. Additionally, projects help students integrate concepts learned in the course by providing an opportunity to apply their skills acquired to work on real-world problems. The students are initially presented the computational algorithm in class in a pseudo-code format. Note that the students do not have any prior exposure to the computational algorithms assigned. However, our experience indicates that the students are able to grasp the computational steps involved even though they do not necessarily have an in-depth understanding of the algorithm. The opportunities for fine grained parallelism are then pointed out by doing an analysis of data flow in the algorithm. For different parallelization strategies, the students are made aware of the computing, communication, and synchronization trade-offs as applicable to computing on FPGAs. Students then work either individually or in groups of two over the remainder of the semester in implementing the algorithm on FPGA. The instructors and teaching assistants are available to help the students on any problems they have. The students are required to make a short presentation of the project on the last day of class. A brief description of some of the projects assigned to our students over the past two semesters is given below.

1. Edge Detection (Image Processing)
   Description: Edge detection is a terminology in image processing and computer vision, particularly in the areas of feature detection and feature extraction, to refer to algorithms which aim at identifying points in a digital image at which the image brightness changes sharply or more formally has discontinuities.
   Deliverables: FPGA implementation of the Sobel edge detection algorithm. An input test image and the algorithm pseudo code are provided.

2. Gauss-Seidel Powerflow computing (Scientific Computing)

Description: Gauss-Seidel algorithm is used in calculation of bus voltages in a power grid.
Deliverables: FPGA implementation of the pipelined PV bus voltage calculation. One group implements the trigonometric computations and the second group implements the remaining computational elements of the PV bus voltage calculation. Input test data and the algorithm pseudo code are provided.

3. Needleman-Wunsch sequence matching algorithm (Bioinformatics)
Description: Needleman-Wunsch algorithm is used for gene and protein sequence matching to determine global alignments.
Deliverables: FPGA implementation of the pipelined Needleman-Wunsch algorithm. Input test sequences and the algorithm pseudo code are provided.

4. LU Decomposition (Scientific Computing)
Description: The LU decomposition is used to generate lower and upper triangular matrices which in turn allow solving of a linear system of equations.
Deliverables: FPGA implementation of the LU decomposition algorithm. An input test matrix and the algorithm pseudo code are provided.

Assessment

We have instituted a rigorous process of measuring our effectiveness in achieving the course goals as well as in ensuring that the relevant ABET criteria are met. A pre/post evaluation test was jointly developed by the authors and administered to the students at the beginning and end of each semester. The test helps us identify students' pre-requisite knowledge and serves as a baseline for measuring knowledge acquired as a result of the course. Grades on end-of-semester projects are used to evaluate students' ability to successfully apply concepts. Recently, we have introduced pre-project and an identical post-project tests administered before and after the final project to determine its efficacy in enabling student learning of course concepts (see Appendix 1). In spring 2009, external evaluators will be used to assess student projects. One of the authors (who also provides leadership for a wide variety of assessments that support ABET accreditation and strategic planning) conducts a focus group (see Appendix 2) at mid-semester to solicit feedback from the students regarding opportunities for improvement. Some of the students' suggestions are implemented prior to the end of the semester; others are implemented the following semester. During the focus group, the instructors are not present so that students feel comfortable in sharing their opinions. Results of the focus group are summarized and reported in aggregate to ensure anonymity. Also, the project investigators and the teaching assistants meet regularly during the semester to discuss opportunities for improving the course and strategies for implementation. Assessment results suggest that the new course provides our computer engineering seniors with a good introduction to reconfigurable computing using FPGAs.

Pre/Post Evaluation test

The pre/post evaluation test consists of a total of seven questions with two questions testing student knowledge of pre-requisite topics, and five questions testing topics covered in the course. Four additional items related to "soft skills" were included on the test. Students are asked to rate

their level of confidence in their ability to: (1) design and implement projects to satisfy performance expectations; (2) design and implement projects on time; and (3) make professional presentations. The last question asked students if their most recent team experience in another course was a positive one. These "soft skill" questions are included on the pre/post tests so that the project team could determine if the students needed additional instruction in these areas.

All results are reported for the Fall semester of 2007. At the beginning of the semester, 79% of the students were "somewhat" or "very" confident in their ability to design and implement projects to satisfy design performance; 79% believed they could complete projects on time; 86% were confident in their ability to make professional presentations; and 86% indicated that their last team experience was a positive one. These findings suggested that students had sufficient opportunities to practice the soft skills prior to ECGR 4146. Consequently, the topics were addressed as needed during lecture and labs and in consultation with students via email, telephone, and office hours. Data for the technical test questions were screened prior to analysis.

Table 1 provides pre- and post-test descriptive statistics. The departure from normality based on skew and kurtosis was clearly due to outlier responses. Deletion of outliers is a generally accepted practice for normalizing distributions, however, given the small sample size (n = 11), they were retained for further analysis.

Table 1

Comparison of Pre- and Post-Test Performance
(n =11)

	Max. Points	Pre-Test				Post-Test			
		Mean	SD	Skew	Kurtosis	Mean	SD	Skew	Kurtosis
Q1*	10	1.73	3.3	2.2	4.1	6.27	3.1	.12	-1.8
Q2	5	0.18	0.6	3.3	11.0	3.55	1.4	.01	-2.2
Q3*	5	0.45	1.5	3.3	11.0	4.00	1.8	-1.5	1.1
Q4	5	0.00	0.0	N/A	N/A	3.91	1.2	-.60	-1.3
Q5	10	0.36	0.7	1.8	2.6	4.91	2.7	.79	-.13
Q6	10	0.82	1.3	1.7	2.3	4.18	2.8	.31	1.2
Q7	5	1.09	1.5	1.9	4.3	2.73	2.1	-.16	-1.8
Total	50	4.64	5.9	1.6	1.5	29.55	7.9	.60	-1.1

** These questions were related to material learned in the pre-requisite course, Advanced Digital Logic System Design.*

A dependent t test was conducted with $\alpha = .05$ to determine if there was a significant difference between students' pre-test performance (M = 4.64, SD = 5.9) and their post-test performance (M = 29.55, SD = 7.9).Results revealed a significant gain in knowledge (t = 10.3, $p < .01$, df = 10) during the semester, which was expected as students had virtually no knowledge of FPGA concepts prior to taking the course. Effect size was calculated using original standard deviations rather than pooled standard deviations as the latter overestimates the effect for dependent pairs. Cohen's d = 3.2, which indicates that the gain in knowledge was very large, i.e. equivalent to more than three standard deviations.

In summary, based on the sample of students enrolled in ECGR 4146 during the Fall 2007 semester, the course significantly contributed to their knowledge of logic design, VHDL and FPGA design flow. However, the results strongly suggested that students may not possess the pre-requisite knowledge necessary for success in the course. Two actions were taken to remedy this situation in Spring 2008. First, this feedback was shared with the Advanced Logic System Design faculty. Second, a brief review was incorporated at the beginning of the semester as a lecture and a homework assignment. With these remedial actions, in subsequent semesters, we have noticed an improvement in student performance on the pre-tests on questions related to the pre-requisite material.

Focus groups

Results of the focus groups also provided valuable insights. For example, during the first semester that the course was offered, students requested that additional materials be available on the course website and that teaching assistants be more readily available in the labs. In response, the instructors enhanced the content of the course webpage and increased the number of teaching assistant contact hours. Similarly, in the second semester, based on the feedback from the previous semester, projects were assigned earlier in the semester with intermediate milestones to provide students with more time to complete the project. Also, based on the student input, the variety of the projects was increased. A follow up focus group meeting revealed that the students greatly appreciated the changes made. Subsequent focus groups in the following semesters provided additional opportunities for minor refinements to the course.

Conclusions

With the rapid emergence of diverse computing architectures in recent years, we were motivated to introduce to the undergraduate curriculum topics such as reconfigurable computing traditionally taught at the graduate level. Also, with the proliferation of multicore architectures the skills for writing parallel software needs to be developed in our students for their success in their future engineering careers. The new course aims to address these goals by exposing computer engineering seniors to a variety of topics including hardware design using VHDL, FPGA design flow and interfacing, data parallel algorithms, and high performance computing applications. After having taught the new course three times over the past two years, we believe that such a course can be taught successfully in many of the nations' computer engineering programs albeit with a lot of effort from the instructor and teaching assistants. Many students reported that four weeks to cover VHDL was too short. However, they added that the integration of the laboratory and the lectures made their learning a lot easier. Here we note that having a skilled teaching assistant, preferably a graduate student with research interests in reconfigurable computing is of a great help. Nevertheless, to better prepare students for this course, we are currently working with faculty from the pre-requisite course Advanced Digital Logic Systems, to give students an earlier exposure to behavioral design and synthesis in VHDL. The students also were challenged with the concept of exploiting data parallelism in sequential algorithms and reformulating the sequential algorithm as a parallel algorithm. The use of familiar examples such

as 1D discrete time domain convolution and dense matrix multiplication greatly helped the student in the understanding process. Although the students found the high performance computing course projects exciting, given the steep learning curve involved in starting from a serial pseudo-code algorithm to developing a fully functional hardware implementation on an FPGA within a month's time, the availability of the course instructor in helping the students with the project is critical. The post-project test and informal interaction with the students at the end of the semester revealed that the project had played a vital role in integrating course concepts.

Our overall conclusion is that with a carefully planned syllabus, course projects, and the availability of student support resources, introducing reconfigurable computing to undergraduate computer engineering students can be a useful vehicle for teaching topics on parallel hardware and parallel algorithms. We plan to make available online all the course materials developed for the new course.

Bibliography

1. Douglass, S., "Introducing the Virtex-5 FPGA family", Xcell Journal, pp. 8 -11, Fourth Quarter, 2006

2. Pellerin, D., and Saini, M., "FPGAs provide acceleration for software algorithms", FPGA and Programmable Logic Journal, Volume: 2, Number 3, January 2004.

3. http://www.coe.uncc.edu/~amukherj/INTRO2VHDL/index.html

4. P.J. Ashenden, The Designer's Guide to VHDL, Morgan Kaufmann, 1996.

5. M.B. Gokhale, C.D. Rickett, J.L. Tripp, C.S. Hsu, and R. Scrofano, "Promises and Pitfalls of Reconfigurable Supercomputing," Proc. 2006 Conf. Eng. of Reconfigurable Systems and Algorithms, CSREA Press, 2006, pp. 11-20;

6. http://www.impluse.com

7. http://www.mitrion.com

8. www.xilinx.com

9. D. J. Evans, "Systolic Algorithms", Gordon and Breach Science Publishers, 1991.

Appendix1: Pre-project test / Post-project test

ABET Learning Outcomes: a, b

1. Write a behavioral VHDL code for the T flip-flop. *(Course Objective 1)*

2. (a) Design a 3-bit ripple counter using 3 T flip-flops. *(Pre-requisite knowledge)*

(b) Write a structural VHDL code of the above counter, with parallel loading capability.

ABET Learning Outcomes: a, h

3. (a) Briefly discuss a possible application of the counter in either a scientific computing, or a bioinformatics processing unit, or a signal processing hardware. *(Course Objective 3)*

(b) Discuss the impact of dedicated hardware processing on the application areas listed above. *(Course Objective 3)*

ABET Learning Outcomes: c, e

4. A digital filter is implemented by the following transfer function

$y[n] = x[n] + 2x[n-1] + x[n-2] – 0.25y[n-1]$ where $x[n]$ input and $x[n-p]$ are the inputs p time steps past. Similarly $y[n]$ is the present output and $y[n-p]$ is the output p time steps past.

(a) Design a digital system capable of implementing this filter. Identify the individual logic elements of your design. *(Course Objective 4)*

(b) Calculate the latency and throughput of this system. *(Course Objective 4)*

ABET Learning Outcomes: c,e

5. Two input 3x3 matrices A and B are multiplied to give an output matrix C using the following algorithm. Note all matrices A, B, and C have 9 elements each.

$Cij = \sum Aik*Bkj$, $k = 1, 2, 3$

For example, $C_{11} = A_{11}*B_{11} + A_{12}*B_{21} + A_{13}*B_{31}$

(a) Design a pipelined implementation for a 3x3 matrix multiplier, assuming you are given three multiply-add (MAC) blocks where each block is capable multiplying two numbers and adding two numbers. *(Course Objective 4)*

(b) If each block takes N cycles to perform its operations, calculate the latency and throughput of the system (expressed in cycles) *(Course Objective 4)*

(c) How many cycles are required for calculating the final output matrix C? *(Course Objective 4)*

ABET Learning Outcomes: k

6. (a) Draw a flow chart showing a typical FPGA design flow. *(Course Objective 2)*

(b) List some of the possible ways to interface an FPGA to a PC. *(Course Objective 2)*

Appendix 2: Sample focus group questions

1. How well do you feel your previous courses prepared you for success in this course?
    a. Which course(s) in particular were helpful in preparing you for this course?
    b. Are there other courses that you should have taken to better prepare you for this class?
2. How would you describe the pace of this class relative to your other ECE courses?
3. How helpful has the textbook for this course been?
    a. What other resources have you used to help you learn this material?
    b. Which resources have been especially helpful
4. How have lab experiences helped you learn and apply the material taught in lecture?
5. How well do you understand and think you can apply the concepts of
    - Pipelining?
    - FPGA design flow?

6. Is there any specific content or material that you are having difficulty with? If so, which one(s)?