The Integrated Computer Engineering Design (ICED) Curriculum

Augustus K. Uht
Department of Electrical and Computer Engineering
University of Rhode Island
Kelley Hall
4 East Alumni Ave.
Kingston, RI 02881-0805, USA

(401) 874-5431
`uht@ele.uri.edu`
`www.ele.uri.edu/~uht`

Abstract

A new undergraduate computer engineering curriculum, ICED, is being introduced at the University of Rhode Island. The main feature of the curriculum is a design project spanning the last three years of the major. This gives continuity to a student's studies: they will always know why they are learning a particular topic, and how it fits into the big picture. It also introduces them to long-term projects, and the requisite good documentation and communication habits necessary for its completion.

The project to be undertaken is the design, simulation and construction of a computer and its compiler, including the design of its instruction set. Further, the various students' computers will be networked together during the final integration phase of the project. Thus, several aspects of computer architecture are treated in depth. Students will learn to make hardware/software design tradeoffs, as well as get hands-on experience with hardware.

A key element of the design experience is the use of modern CAD tools. The Mentor Graphics CAD tool suite will be used throughout the curriculum. By standardizing on one set of tools, the time for the students to learn the tools is amortized over the entire curriculum.

The curriculum has received funding from the National Science Foundation, and has been formally approved by the University. Students began the curriculum in Fall, 1997.

This paper gives the rationale of ICED in depth, and describes the core courses, their activities and use of CAD tools, and how they interrelate to achieve the goals. ICED's progress is also reviewed.

1 Introduction

The University of Rhode Island has developed a novel undergraduate Computer Engineering (CE) major curriculum. Traditional CE curricula are composed of disparate courses which poorly unify key themes and do not allow students to develop critical abilities. The Electrical and Computer Engineering (ELE) department at URI will remedy these ills with a new CE curriculum. The centerpiece of the curriculum will be the design of a richly complex digital system: a network of computers including, particularly, a computer with hardware and important system software built

This work was supported in part by the National Science Foundation through grant number DUE-9751215, and in part with software donations from the Mentor Graphics Corp. and Xilinx Corp.

This is a revision of a paper that appeared in the Proceedings of the 1997 International Mentor Graphics Users' Group Conference.

entirely by each student. This project will span the courses taken for the last 3 years of each undergraduate's studies.

This curriculum will allow a novel emphasis on fundamental engineering understanding, which can not be developed in projects of shorter duration. Students will develop an understanding of how design decisions play out over time, they will employ sophisticated design techniques using standard simulation and synthesis tools and they will make complex software/hardware tradeoffs.

This unified, longitudinal Integrated Computer Engineering Design (ICED) curriculum has been planned over several years. It has been accepted by the university, and began in the Fall 1997 semester.

This paper introduces the ICED curriculum and its goals. Since ICED is in the process of being realized, this paper presents a work in progress. Our recent experience with the first courses in ICED is given, including a description of our initial evaluation methodology.

Curricular Need - The old version of the computer engineering curriculum lacked cohesion, as well as large-systems design, construction and evaluation. Also, there was not a large software component, and hardware and software were not linked in a meaningful way, especially to understand the tradeoffs between the two. Further, the emphasis on networks was not as great as it should be, especially with the explosive growth of the Internet.

The students need experience in complex digital system design, simulation, construction and evaluation. Although design these days is predominantly done with simulation, physical experience with complex hardware is essential for a computer engineer. "...there is something about late night debugging of that 'last' hardware problem that gives student engineers respect for the real difficulties of hardware, that simulations alone do not provide."[1]

Related Curricula - Several schools design and build computers as part of their CE curricula, e.g., Georgia Tech[3, 4], UCSC[10] and others[5]. However, such projects are usually done all at once, as capstone Senior design projects. A key element of ICED is a three-year long in-depth design project as part of most of the CE courses.

Some schools also involve compilers in the capstone projects. At Georgia Tech, this includes the modification of a retargetable C compiler to accommodate the student-designed computer. However, compiler construction in general is not incorporated in the experience; to our knowledge no school integrates a complete compiler design course with the computer hardware component of a CE curriculum. In fact, no school seems to require a compiler construction course as part of its CE curriculum[5]. Therefore, in such traditional curricula, the opportunity to make hardware/software design tradeoffs is relatively restricted, especially considering the large role such issues have in current computer (and most digital system) design. ICED requires a compiler construction course; it is integrated with the major project.

Many excellent tools have been devised to study computer architecture[6]. While they may provide much in the realm of understanding of an existing architecture, they do not give the student much freedom in design, hardware/software tradeoffs are not investigated in detail, and the student does not work on real hardware, only simulations. In [12], a comprehensive simulation environment employing both varying software and hardware components is available to the student; however, the student is only able to change system specifications: the changes to the compiler and hardware are made automatically. This does not give the student any design experience, and forces the student to only use the options allowed by the simulator writers. In our initial experience with

[1]Robert Colwell, Chief Architect of the Intel Pentium Pro, at the panel session at the International Symposium on Computer Architecture, May, 1996 (paraphrased).

allowing the student almost complete freedom in architectural specification and design, we have been amazed at the rich diversity of approaches students have taken in their designs, a refreshing change from the usual vanilla 32-bit RISC CPU.

Another issue involves the design tools used in a curriculum. Although it is possible to construct one's own in-house CAD (Computer-Aided-Design) tools[1], keeping them simple (as compared to industrial tools) and thus easier to learn, the student then does not experience the full power of industrial tools, nor does she/he become acclimated to the complexities of real-world design. Another issue is that at some schools different design tools are used in different courses, adding to the overhead of tool-learning time to the student. In ICED, the industrial CAD tool suite from Mentor Graphics will be used throughout the curriculum.

In current curricula, students do not participate in projects lasting more than one or two semesters. In practice, projects last much longer, often years. With the major project occurring over three years, ICED necessitates that the student create clear documentation, and deal with outcomes beyond any one course.

There are related projects. In [7] and others, networks and distributed systems are investigated, but only from a programming point of view, and the studies are not integrated with other topics. Integrated research across the curriculum is proposed in [11], but this is in the field of Psychology. In a closer but still different area, [9] proposes an integrated curriculum in Digital Signal Processing and Communications. Many hardware simulation, design, and emulation or construction curriculum projects have been pursued, including the use of modern Field Programmable Gate Array chips, such as in [2, 8], but the study of compilers was not included, networks were not addressed, and the work was not integrated across a curriculum.

2 The New CE Curriculum

A key ingredient to successful digital system design is making appropriate design tradeoffs amongst the hardware and software components of the system to achieve a suitably-performing, cost-effective design. The ICED curriculum ties together what is traditionally unrelated content from different courses. This allows students a key, novel experience – the design, over years of studies, of an actual, complete, working network, particularly including a computer with both the processor and the compiler built by the student.

The design tasks to achieve these aims are distributed amongst the required core courses as follows. The courses are taken roughly in the order shown. The catalog descriptions of the courses are given first, then the design tasks are described in *[italics]*, followed by the names of tools used (all of them Mentor, unless noted otherwise).

1. **ELE 201/202 - Digital Circuit Design:** Logic gates, Boolean algebra, combinational and sequential circuits, analysis and design of sequential systems, multi-input system controllers, asynchronous finite state machines. Laboratory experience in digital electronics; logic design projects using standard integrated circuits.

   [Familiarization with industry-standard CAD tools. Design, simulation, construction and test of components for the computer, e.g., an adder, as well as simple control circuits. Introduction to the VHDL hardware modeling language.] Design Architect (schematic capture), Quicksim II (schematic simulation), QuickHDL (VHDL Compilation and Simulation).

2. **ELE 305 - Introduction to Computer Architecture:** Architecture of digital computers. CPU microarchitecture. Instruction execution cycle. Instruction sets. The memory hierarchy. Pipelining, instruction level parallelism, introduction to parallel computing. Register-level design and simulation of a simple computer.

[Design and simulation of a simple machine instruction set for the student's computer, using VHDL and other CAD tools. High-level design is performed.] QuickHDL, Renoir (high-level design and VHDL synthesis, e.g., graphical state machine and truth table entry, and conversion to VHDL).

3. **ELE 405 - Digital Computer Design:** Hardware implementation of digital computers. Arithmetic circuits, memory interface, data path, control path, input/output. Synthesis tools. Gate level design, simulation, construction and verification of a simple digital computer.

   [Design, simulation, construction/emulation and test of the digital hardware realizing the instruction set devised in ELE 305. Gate-level design is performed, along with VHDL aids (e.g., for logic synthesis). The same suite of CAD tools is used. The computer will be emulated, debugged and evaluated, using rapid prototyping hardware (FPGA), and logic analyzers.] Design Architect, QuickHDL, Quicksim II, Renoir, Galileo (VHDL-to-logic synthesis tool); Xilinx M1 (FPGA mapping, routing, and configuration).

4. **CSC 402 - Compiler Design:** Grammars and languages; lexical analysis, parsing and translation, symbol tables, runtime storage administration, object code generation. Students will construct a compiler for a small programming language.

   [A compiler providing translation from a high-level programming language to the machine instruction set of the student's computer is written and tested.] HP logic analyzer source-code–to–analyzer mapper.

5. **ELE 437 - Computer Communications:** Computer networks, layering standards, communication fundamentals, error detection and recovery, queuing and delay-thruput trade-offs in networks, multiple-access channels, design issues in wide and local area networks.

   [A simple wireless or wired local area network is designed, incorporating standard network elements such as layering, error detection and recovery, and throughput management.] Design Architect, Quicksim II, QuickHDL, Renoir.

6. **ELE 408 - Computer Systems Laboratory:** Engineering design problems involving modern microprocessor systems, operation of ALUs, data paths, control units, input/output, and memory systems.

   [The hardware and software of the computer are integrated, and the overall system is debugged and evaluated using benchmark programs. The effect of hardware/software design modifications on cost and performance is measured. The different computers of the entire class are hooked together in a simple network; communication amongst them is achieved. The network is evaluated. The emulation hardware and logic analyzers are again used.] All of the above tools.

Other optional or potential components (these are not initially required, to ease the realization of ICED):

- **CSC 412 - Operating Systems:** Presentation of the general concepts underlying operating systems. Topics include process management, concurrency, scheduling, memory management, information management, protection and security, modeling and performance.

  [A simple operating system / monitor is written and tested for the computer.] HP logic analyzer source-code–to–analyzer mapper.

- **ELE 447 - VLSI Design and Simulation:** Design and simulation of digital integrated circuits. Extensive use of software tools such as Mentor Graphics' IC Station and simulators. Student designs are fabricated and tested.

  [The computer hardware, or parts thereof, are realized again, this time by designing a custom Integrated Circuit (IC). The IC is fabricated via MOSIS.] IC Station, Lsim, etc.

(The complete curriculum is available via the author's web site: `www.ele.uri.edu/~uht`)

Thus, the students are exposed to many industrial CAD tools in the course of their studies, including schematic capture (Design Architect), hardware programming language use (VHDL, via the QuickHDL tools), high-level design (Renoir), and synthesis (Galileo). With the guidance of the professor, students estimate the consequences on the compiler design when designing the hardware in ELE 305 and 405. During compiler construction in CSC 402, in their Senior year, students may discover that they need to change the balance between hardware and software, and modify their hardware designs accordingly. The network component is addressed in ELE 437. Lastly, in ELE 408, all of the pieces are put together. Experiments on hardware/software tradeoffs are performed, and the network is tested.

Equipment - The Mentor suite was chosen as the primary CAD tool provider in the department after an exhaustive evaluation and comparison of the major CAD tool vendors. Mentor was the clear favorite, being the most complete and lowest cost of the vendors' offerings. It is also very popular in industry, and may legally be used in research (both undergraduate and graduate), especially that involving industrial participation.

In April of 1997 we received an equipment grant from the National Science Foundation for the support of ICED. With the grant, we have purchased 10 Sun Ultra 1's with 128 MB RAM to run the Mentor tools and the emulation hardware. We are also using our old Sun IPC's as X terminals, one IPC connected to each Ultra, to stretch the number of workstation seats further. We also plan to purchase a Sun compute server, for jobs too large for the Ultra's.

Our lab setup will be based on commercially available rapid prototyping emulation hardware units employing Field Programmable Gate Arrays (FPGA's); they are Virtual Computer Corp. EVC1s units. The programmability of the FPGA's allows for many experiments of different computer designs to be made. Emulation[2] is being used more and more in industry, both for final products as well as prototyping, so this will give the student valuable exposure to a modern technology, lacking in the old curriculum. Discrete components will be used for the network interface, adding to the "hands-on" experience. Three lab stations have been setup, each with its own emulation unit, HP 1662CS logic analyzer and Sun Ultra (out of the 10 above); a prototyping board extension to the EVC1s unit will be added in the near future. Given our current class size (about 30 students in ELE 405 in the Spring 1998 semester), this might seem to be an insufficient number of stations. However, all of the design work can be done on any of our Sun's; a full lab station is needed only at the end of a lab for physical debug and cost and performance investigations.

We will have some interesting capabilities with the lab equipment, once all of the software tools have been installed. In particular, it will be possible to correlate an analog view of a signal with a digital view of the signal, with its particular machine/assembly instruction, and finally with the corresponding original high-level source instruction of the test program. This will give the student the ability to see the effect of a design change at one of these levels on all of the levels, and thereby see the effects of hardware/software tradeoffs in an extremely concrete way.

[2] "Emulation" is perhaps a misnomer in this case, since the logic of the computer IS realized in physical gates and storage.

3 Progress Report

As stated above, the lab equipment is mostly in place.

ELE 305 was taught for the first time in the Fall 1997 term. The class contained about 60% computer engineering majors with the remainder being electrical engineering majors (ELE 305 is required for all of the department's majors; the following course, ELE 405, is only required for computer engineers). It had mixed success.

For the main project, the class was divided into about 8 teams of about 4 students each to create their instruction sets, model and evaluate them, and model a complete computer system with CPU, bus, memory, and I/O. Each team was composed of an architect, a designer, a validator, and a recorder; over the course of the different parts of the project, each student was to have each role once. Each team produced a single report for a project. A student's grade on a project was the mean of the grade on the individual student's contribution to the whole effort, and an overall team grade.

Bare VHDL was used for the first, simple designs. The CPU and system were created with the high-level Renoir design tool; this allows the direct graphical entry of state machines, truth tables and block diagrams. Renoir generates VHDL code from this input. All of the VHDL models were to be simulated and verified with QuickHDL.

At the end of the term, only 1 or 2 teams had their computer systems substantially working, but most teams were close (everything designed, much of it debugged). The team structure did not work consistently. In many teams, the roles were not actually rotated; the same people tended to do the same tasks throughout the term; there were a couple of cases of half of a team doing almost all of the work, and the other half doing little. This was especially true at the end of the term. Although the students' course and instructor evaluations have not been processed, even with the above problems, the informal feedback I have gotten from the students has been very positive.

The students worked very hard, many of them were highly enthusiastic. I believe I erred in requiring them to learn too much on their own, especially with respect to mastering the CAD tools. A lot of time was spent figuring out design methods and tool idiosyncrasies. This also slowed down the lecture pace, as much class time was spent on these issues. I believe they still learned a lot, and that these problems are correctable. I am revising my approach in this term's (Spring 1998) ELE 405 class, and trying to ease their assimilation of new tools to a much greater extent. Further, their first task is to finish their ELE 305 projects, so no one will fall behind, in the long term.

I have largely given up on the big team concept. It is too easy for individual students to slack off, and not learn what they should. Although the conventional wisdom is that schools these days are supposed to develop teamwork skills in their students, it is my view that my role is chiefly to aid the student in acquiring technical knowledge (especially principles), skills, and thinking abilities, not to provide job training. The latter is fine, as long as it doesn't overly impact the former. Thus, in ELE 405 I have returned to the standard undifferentiated 2-student partnership as the basic entity. Also, every student will have to submit their own reports; they have to be able to write. (Partners can of course share data, graphics, etc., but the English must be an individual's own.)

Beginnings of a Formal Evaluation We want to be able to make a concrete judgement of the ongoing success or failure of ICED. We have devised a methodology to do this, which is itself somewhat experimental.

In the original formulation, every student was to fill out a questionnaire asking how satisfied they were with the new or old curriculum. However, this was abandoned since it did not provide any objective evaluation of the student's abilities. The approach we ARE trying is now described.

At the end of each term every student in every core class in the Junior or Senior years is asked to fill out a survey form. This form consists of two parts: data on the student, such as major, class, courses taken to date, etc.; and a hardware/software tradeoff question. (The question used in Fall 1997 is given in Appendix A. Note that there is no single correct answer.) The forms are filled out in class at the same time.

Half of the students (chosen at random) also provide their name in the first part. This was done since some of us felt that without a student putting their name on the sheet, the student would not take answering the question seriously, and would not expend effort on it. We will look at the difference between the two groups to try to resolve this for future surveys. The results of the survey do not affect their grades in any way.

After completion, the two parts of each survey are marked with a unique random number for identification purposes; the parts are then separated. The second parts, the answers to the tradeoff question, are combined from all of the surveyed classes and shuffled. They are then evaluated independently by two or more computer engineering faculty members, using a common grading guide sheet, and without knowledge of the contents of the corresponding first part's information; this includes not knowing the corresponding student's name. Once this data has been generated, it is re-linked with that of the first part, and summarized and analyzed, noting correlations or anti-correlations, and relative abilities of different classes. Note that for this academic year all of the Seniors are on the old curriculum. They are also surveyed in relevant classes, and these forms are also put in the common pile before grading. Therefore, the grading is curriculum, student, class, major, student background, and course blind. However, once graded, all of the variables are separated out, and the correlations can be made. We will also get some information by looking at how different graders graded the same forms.

The results from one semester will not necessarily be indicative of ICED's effectiveness, although they may be. The survey results from successive semesters will be tracked over time, and then summarized.

The survey was given out in the Fall 1997 semester. The forms are still being processed. Time and space constraints permitting, they will be included in the published version of this paper.

4 Summary, Status and Conclusions

For those students desiring hardware and software computer engineering design skills, as well as the underlying theoretical knowledge, to create complex, realistic digital systems, the new URI curriculum will offer a unique experience. We believe that our experiences with this ambitious, new approach to undergraduate curriculum should be a valuable model for programs across the nation.

ICED is becoming a reality; it has been approved, and became official with the Fall 1997 semester. The Mentor donation was granted two years ago (Fall 1995). Since then we have used many of the CAD tools with great success in our courses. The new workstations were installed in the Summer of 1997, and were used in ELE 201/202 and 305 in the Fall. The lab stations have been purchased, and are being configured in time to be used by ELE 405 and 437 in the Spring. Although only Freshman were required to begin the new curriculum this Fall, Sophomores and Juniors were *de facto* switched to the new curriculum at that time. Thus, the first students following ICED will graduate in less than two years.

The ICED curriculum is a novel attempt to bring the problems and opportunities of a long-term project into the undergraduate curriculum, allowing students to engineer complex digital systems from both software and hardware components, and to comprehend the structure of these systems, from networks to gate-level logic in a CPU. Many aspects of computer architecture are investigated, including Instruction Set Architecture, microarchitecture, system architecture, and network architecture. Hardware/software tradeoffs are studied throughout the curriculum.

Acknowledgments

The new ICED curriculum is the result of many peoples' efforts over many years. An initial version was proposed with Laurette Bradley at UCSD in 1987-88. Independently, J.C. Lo began work on something similar at URI in the early '90's. The main effort in realizing the final ICED curriculum comes from the faculty of the Electrical and Computer Engineering Department at URI, especially: G. Sadasiv, J. Daly, G. Fischer, J.C. Lo, W. Ohley, Y. Sun, D. Tufts, and Q. Yang. J. Kowalski of Computer Science is heading the compiler part of the project. Our systems staff people, especially T. Toolan and T. Nightingale, have been critical in making it happen. We have also been aided by Glenn Erickson and Bette Erickson of the URI Instructional Development Program.

References

[1] G. M. Brown and N. Vrana. A Computer Architecture Laboratory Course Using Programmable Logic. *IEEE Transactions on Education*, 38(2):118–125, May 1995.

[2] PI: Cook. Electronic Design Automation Fabrication. *NSF Undergraduate Instrumentation and Laboratory Improvement Awards*, DUE-9452121, July 1994. Via http://www.fastlane.nsf.gov/a6/A6QueryPgm.htm, select "Undergraduate ILI".

[3] Georgia Institute of Technology. GATech CmpE 4510 Computer Engineering Design II. Internet web site, November 1996. http://www.ece.gatech.edu/academic/courses/cmpe4510.html.

[4] J. O. Hamblen, H. Owen, S. Yalamanchili, and B. Dao. Using Rapid Prototyping in Computer Architecture Design Laboratories. *IEEE Computer Architecture Technical Committee Newsletter*, pages 44–52, June 1996. Part of the 2nd Annual Workshop on Computer Architecture Education, held in conjunction with the Second High Performance Computer Architecture Conference.

[5] Kaeli, D. R., editor. *IEEE Computer Architecture Technical Committee Newsletter - Special Issue on Computer Architecture Education.* IEEE Computer Society, June 1996.

[6] Kaeli, D. R., editor. *IEEE Computer Architecture Technical Committee Newsletter - Special Issue on Computer Architecture Education.* IEEE Computer Society, September 1997.

[7] PI: McCabe. A Programming Laboratory for Computer Networks. *NSF Undergraduate Instrumentation and Laboratory Improvement Awards*, DUE-9452037, September 1994. Via http://www.fastlane.nsf.gov/a6/A6QueryPgm.htm, select "Undergraduate ILI".

[8] PI: Perskowski. Use of FPGA Hardware Emulator and VHDL to Teach Digital Design. *NSF Undergraduate Instrumentation and Laboratory Improvement Awards*, DUE-9452593, October 1994. Via http://www.fastlane.nsf.gov/a6/A6QueryPgm.htm, select "Undergraduate ILI".

[9] PI: Thompson. Electrical Engineering Systems Laboratory. *NSF Undergraduate Instrumentation and Laboratory Improvement Awards*, DUE-9552060, July 1995. Via http://www.fastlane.nsf.gov/a6/A6QueryPgm.htm, select "Undergraduate ILI".

[10] A. Varma, L. Kalampoukas, D. Stiliadis, and Q. Jacobson. CPU Design Kit: An Instructional Prototyping Platform for Teaching Processor Design. *IEEE Computer Architecture Technical*

Committee Newsletter, pages 23–26, June 1996. Part of the 2nd Annual Workshop on Computer Architecture Education, held in conjunction with the Second High Performance Computer Architecture Conference.

[11] PI: Volckmann. Research Across the Curriculum in Psychology. *NSF Undergraduate Instrumentation and Laboratory Improvement Awards*, DUE-9452253, September 1994. Via http://www.fastlane.nsf.gov/a6/A6QueryPgm.htm, select "Undergraduate ILI".

[12] Y. Zhang and G. B. III Adams. An Interactive, Visual Simulator for the DLX Pipeline. *IEEE Computer Architecture Technical Committee Newsletter*, pages 9–12, September 1997. Part of the 3rd Annual Workshop on Computer Architecture Education, held in conjunction with the Third High Performance Computer Architecture Conference.

Appendix A. Fall 1997 Tradeoff Survey Question

1. You are an engineer in the GE Whiz Co., designing their next product, a new type of industrial robot. Your specific immediate task is to determine whether one of the necessary functions, an Analog-to-Digital Conversion (ADC), should be realized in hardware or software. Only a few (less than a hundred) of these robots will be sold since they're very expensive. As part of the sensor data processing, the ADC has to be performed continuously during the robot's operation.

   If the ADC is realized in hardware, it will be able to operate at a much higher conversion rate than a software-based version, allowing the robot to sense its environment more frequently, and thus move faster and perform its tasks faster. The hardware ADC is expensive, costing about 5% of the total cost of the robot.

   A software version of the ADC would cost much less. It would run on the robot's main processor.

   It is expected that over the lifetime of an individual robot, it is likely that both hardware and software versions of the ADC could be improved, making the product more attractive to potential buyers.

   Consider all aspects in answering the following question, including cost, performance, functionality, etc.

   QUESTION: Which do you recommend, using the hardware or software version of the ADC? What considerations led you to your recommendation?

2. Your boss liked your work on the big robot, and now wants you to realize the same ADC function in a new small, cheap, personal robot. The marketing people say a million or more can be sold.

   QUESTION: Which do you recommend, using the hardware or software version of the ADC? What considerations led you to your recommendation?