BUILDING REAL COMPUTER SYSTEMS

THE AUTHORS DESCRIBE A MULTIYEAR STUDENT PROJECT ON THE DESIGN AND CONSTRUCTION OF HARDWARE AND SOFTWARE FOR A COMPLETE COMPUTER SYSTEM. STUDENTS USE AND BUILD UPON THESE SYSTEMS OVER THEIR UNDERGRADUATE YEARS.

Augustus K. Uht
Jien-Chung Lo
Ying Sun
James C. Daly
James Kowalski
University of Rhode Island

Engineering products are composite systems embodying contributions from many technical specialties. Thus, it is imperative to expose students to links between various subjects taught separately in a curriculum and to familiarize students with making design choices across technical realms. Computer architecture requires a systems approach, that is, the ability to make choices in general and hardware and software tradeoffs in particular. However, constructing complete systems is so enormously complex that it is hard to teach, and, indeed, is usually not taught. Instead, students often build academic point-solution systems, which are implementations of computer architectures that they will never use. But the discipline of building for later use teaches students to place a high priority on quality, documentation, and robustness in unforeseen environments. These habits make students more attractive to later employers and are also fundamentally useful mind and work habits.

At the University of Rhode Island (URI), we developed the Integrated Computer Engineering Design (ICED)[1-2] curriculum to give students the vital experience of building systems for real use. The key component of ICED is a two- to three-year project that primarily spans the undergraduate junior and senior years. The project entails design and construction of a complete computer system (including CPU, memory, and compiler) networked to other students' systems.

By forcing students to think about the entire system, we concretely establish principles of computer engineering in the students' minds. Concurrently, students learn detailed skills using industrial electronic design automation (EDA) tools and techniques, so students can readily practice their craft in industry. Our goal is to teach students principles for a lifetime and detailed skills for their immediate careers.

Goals

Undergraduate computer engineering programs should school students in the central principles of digital communications and processing in a broad sense, and develop students' abilities to think critically and creatively. This must occur in four short years simultaneously with many other conflicting demands on a student's time.

Computer architecture education should give students a time-honored classic engineering education, while ensuring that they learn timely skills, go into industry and research, and quickly contribute to their profession. Another goal is imparting the ability to make skillful hardware and software tradeoffs. This is of paramount importance today

0272-1732/00/$10.00 © 2000 IEEE

especially in mobile and embedded systems where power, size, and reliability considerations are as important, if not more important, than classic performance and cost issues. Embedded systems account for many more of the world's processors than do traditional computers. Such systems also require architects and engineers to have knowledge of many fields, such as analog electronics and probability (seen in the ICED laboratory in the ICED network and its operation), not just digital hardware and software.

The ICED program's strength is that it integrates classic engineering studies with the timely skills needed for current systems. By focusing on one set of design tools, we can distribute the learning time over several years, keeping total job training time low and letting faculty emphasize principles. The project's primary function is to reinforce principles in students' minds by making these principles come to life in hardware and software. The project's long-term time period enables this function because students always perceive where the current topic fits in the overall plan.

ICED curriculum

The ICED curriculum (see the sidebar for the project's history) uses the CPU as the primary vehicle for hardware design for two reasons. If students can design a CPU, they can design any digital hardware. It's important for computer engineering students to understand how a basic general-purpose CPU works before designing more specialized processors, such as digital signal processors or complex network devices (routers).

There are currently six core courses for the project,

- digital circuit design (sophomore year),
- computer architecture (junior year),
- digital computer design (junior year),
- computer networks (junior year),
- compiler design (senior year), and
- computer system lab (senior year).

Students make hardware and software design decisions throughout the construction of their systems. We initially teach trade-offs directly, and as students progress they learn them through actual experience. Once a system is built, we ask students to meet a new design requirement for their overall system, which requires proposing candidate solutions, choosing a suitable candidate, making design changes, and evaluating the solution.

Students need experience in complex digital system design, simulation, construction, and evaluation. Although currently design is predominantly performed with simulation, physical experience with complex hardware is essential for a computer engineer. As Robert Colwell, chief architect of the Intel Pentium Pro, said at a panel session of the International Symposium on Computer Architecture in 1996, 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 (paraphrased).

Related curricula

Several schools[3] design and build computers as part of their computer engineering curricula including Georgia Tech[4] (with field-programmable gate arrays, or FPGAs). However, such projects are usually completed all at one time, as capstone senior design projects.

We know of no other school that requires a compiler construction course as part of its computer engineering curriculum,[3] although Georgia Tech uses a retargetable C compiler in its student project. While faculty at various schools have devised many excellent tools to study computer architecture[5]—many of which provide much in the understanding of an existing architecture—they do not give students a lot of freedom in design, trade-offs, and real systems. Also, many schools construct their own in-house EDA tools[6] instead of using industrial applications. This keeps the design tools simple and

ICED's history

ICED was originally proposed at the University of California, San Diego in 1987-1988 by Uht and Laurette Bradley (now at GTE), but was not implemented. The initial goal of ICED was to teach hardware and software trade-offs by having students design and build both a compiler and a CPU. Independently, in the early 1990s, Lo investigated the concept of teaching across the curriculum for computer engineering at the University of Rhode Island, with the goal of unifying and relating course content throughout the curriculum. Uht and Angaraih Sadasiv (of URI) obtained funding for ICED in 1997 from the National Science Foundation, and ICED began operation that year. Further funding was obtained from the Champlin Foundations in 1999. We have purchased or built most of the laboratory equipment; some hardware testing and software construction remain.

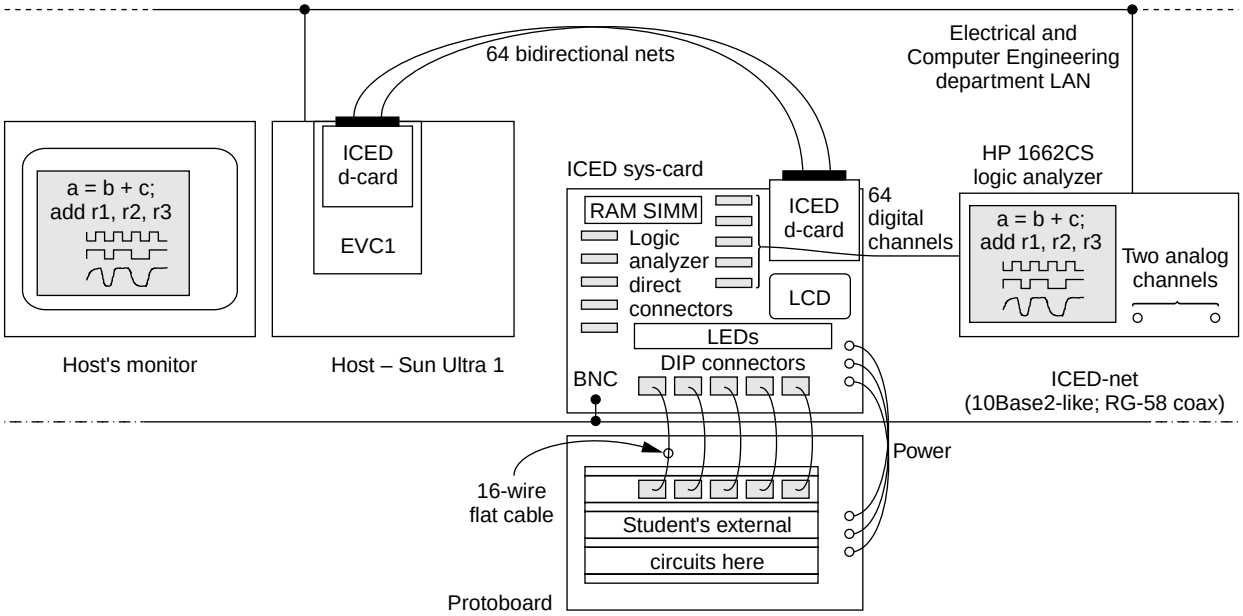

Figure 1. ICED prototype system and lab station. The student CPU is in the FPGA on the EVC1 card. The logic analyzer is completely viewable on and controllable from the host.

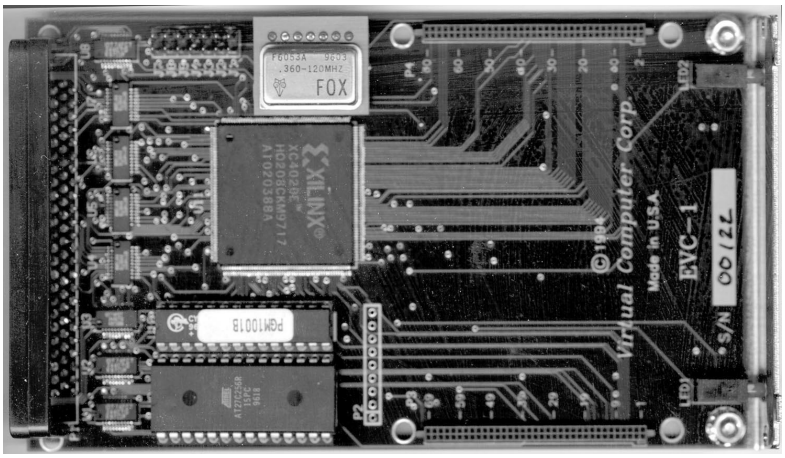

Figure 2. Virtual Computer Corporation's EVC1. From left to right, the main components are the Sun Sbus connector (on obverse, beneath handle), interface and FPGA configuration logic, Xilinx XC4020E-2 20,000-gate-equivalent FPGA, programmable oscillator module (marked "FOX"), and space and connectors (P3 and P4) for a daughtercard (ICED d-card). A retooled back plate (not shown) holds two connectors providing external access to the d-card, EVC1, and hence the FPGA. *(Photo courtesy of Virtual Computer Corp.,*

easier to learn, but students do not experience the full power of industrial tools or become acclimated to real-world design complexities.

Laboratory tools

Our overall laboratory goal is multiple students' computers communicating with each other over an Ethernet-like network. During system test and operation, a logic analyzer lets students see a vertical slice through the design levels of the machine. This view includes source code, assembly code, machine code, digital signals on the computer bus, and analog pictures of particular digital signals in the system.

The vast majority of the ICED laboratories include the design or use of a CPU within an FPGA, which consists of many uniform logic cells wired together via an interconnection network. The FPGA's configuration consists of the specific logic functions realized by the cells and the signal routing and connections, all held in static RAM. We can reconfigure FPGAs an unlimited number of times.

We chose Xilinx's FPGA because it is very flexible and efficient. We use Virtual Computer Corporation's EVC1 card for the FPGA's platform. The EVC1 provides us with a very simple interface to the FPGA. It also saves us a lot of time and expense, since it comes complete with much of the overhead hardware and support software.

Prototype system laboratory

Figure 1 shows a schematic block diagram of the physical components of the lab.

The EVC1 card is housed in the chassis of a Sun Ultra 1 and is connected to one of the Sun Sbus I/O connectors. We make an external connection to the EVC1 through the back panel of the Sun via a custom connector and daughtercard assembly called the d-card. The d-card provides connections to another custom card, similar to a motherboard, called the sys-card. The sys-card contains the student prototype's bus and supporting hardware, including the prototype's main memory. In turn, the sys-card is connected through short cables to a generic protoboard, which holds the student-designed ICED system interface logic. The sys-card also provides a standard bayonet-nut-couple connector tap onto the prototype network, ICEDnet. Lastly, we use a Hewlett-Packard logic analyzer to test, debug, measure, and otherwise investigate the prototype. The logic analyzer has 64 data-input channels, viewable as state or timing information, as well as analog signal and assembly code viewing options.

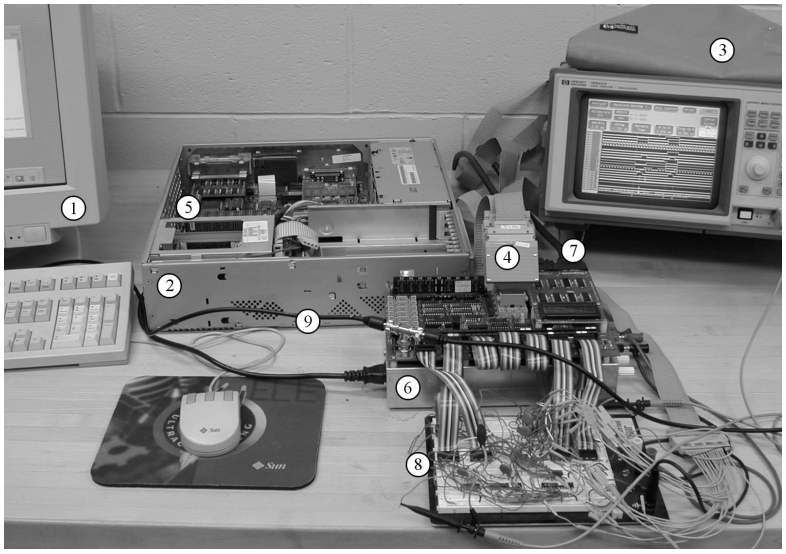

Figure 3. ICED lab station. *(Photo by Laurette Bradley.)*

EVC1 FPGA platform

The major component of the EVC1 (see Figure 2) is the Xilinx XC4020E-2 FPGA, nominally equivalent to 20,000 simple gates. The FPGA is large enough to hold a simple 32-bit processor and can also accommodate 32-bit pipelined CPUs with forwarding at the cost of extra time to route the FPGA's more complex internal interconnections.

The EVC1 has many useful features. First, there is a general-purpose interface from the host's Sbus to the FPGA. In fact, part of the interface extends to within the FPGA and is logically connected to student designs via the EDA tools. Further, students can access many Virtual Computer Corporation-donated C software utilities for such purposes as downloading design files (configuring the FPGA), changing the frequency of a separate clock oscillator (the programmable oscillator module, see Figure 2), general communication with the FPGA, and resetting and testing the EVC1.

We make external connections to the EVC1 via its standard daughtercard connectors. These provide access to general I/O pins—configurable as input, output, or bidirectional—on the FPGA. An ICED d-card containing SCSI-type transceivers plugs into the EVC1 and sends 64 directionally configurable signals to a d-card that is piggybacked on the sys-card. Programmable outputs on the FPGA set the directions of the signals leaving the EVC1's d-card. On the sys-card, the signals are also directionally configurable, this time by jumpers or logic on the student's protoboard.

ICED Protosys laboratory station

Figure 3 shows the overall ICED hardware laboratory setup. The different pieces of equipment are as follows:

1. host computer display;
2. host computer, shown with cover off;
3. logic analyzer with 64 digital inputs and two analog inputs;
4. logic analyzer digital probe with 16 inputs;
5. EVC1 card with FPGA and ICED daughtercard (d-card);
6. ICED system card and chassis (sys-card);
7. ICED computer bus (Ibus) connects d-card on EVC1 with d-card on sys-card;
8. protoboard holding memory, network, and LCD to CPU interface logic; and
9. ICED network (ICEDnet).

We use the Protosys equipment as follows. The host computer runs the equivalent of a monitor program, allowing it to control the Ibus and CPU. With these capabilities, we use the host to download code and data to the ICED computer's memory and then start, stop, or single-cycle the ICED computer. We

> Our design goals for the physical architecture of the Protosys custom hardware were to minimize student wiring drudgery, while maximizing design flexibility and pedagogical impact.

also use the host independently of the Protosys to run Mentor Graphics and Xilinx EDA tools to enter the CPU and other digital designs and configure them for the FPGA. We download the configuration data to the FPGA to create an ICED CPU. We also develop software on the host, including compiler and monitor development.

The custom hardware consists of the d-cards and the sys-card. The d-cards connect the FPGA (ICED CPU) to the sys-card. The sys-card is essentially a realization of the Ibus and the ICED computer proper with multiple, built-in probing points and great interconnection flexibility. The protoboard connects to the sys-card and holds the student-designed interface logic connecting the ICED computer's main memory and I/O devices to the Ibus. Students have access to generic array logic (reprogrammable logic devices) and standard SSI and MSI ICs for this hardware.

The logic analyzer attaches directly to the sys-card through special connectors, reducing the wiring drudgery for students. The analyzer both debugs the ICED computer and visualizes its operation at multiple abstraction levels. X Windows and Internet interfaces on the logic analyzer let users view and control the analyzer from the host's display. In the future, this may also allow some form of remote access and open up distance-learning opportunities with the Protosys station.

With this compact yet sophisticated setup students can make interdependent software and hardware changes, and readily see their effect.

Custom hardware design goals. Our design goals for the physical architecture of the Protosys custom hardware were to minimize student wiring drudgery, while maximizing design flexibility and pedagogical impact. We also wanted to use standardized components to reduce our costs and to give students additional exposure to real-world systems. From the systems perspective, we wanted students to focus on the functional interaction of all of the system components. Lastly, we wanted a system resistant to unintentional student errors, such as driving connected buffers to opposite values.

We achieved these goals. We can connect the Ibus's 32-bit data bus and 24-bit address bus directly to the memory via several dual-inline package (DIP) connectors, bypassing the protoboard. There are address multiplexers for the memory, which reduces wiring. We provide special connectors to allow easy multiple connections (16) to the logic analyzer, in most cases. We use a standard dynamic RAM card. Students have access to all control signals and must use them correctly if a system is to work. Students must design all of the interfaces: Ibus to memory, Ibus to LCD display, Ibus to ICEDnet, and so on. Although the 64-bidirectional signals connecting the d-cards are nominally configured as the standard ICED Ibus, students may arbitrarily reconfigure them to suit particular requirements. The d-card Ibus transceivers we selected can withstand driving each other in opposite directions. They also have built-in thermal protection shutdown circuitry.

We chose a standard dynamic RAM SIMM (single in-line memory module) to expose students to the system and interface problems and principles of memory refresh. The SIMM interface design also introduces students to row address strobe and column address strobe memory device selection. While these features may not concern users of today's high-end systems (Rambus), they certainly concern the memory system designers themselves.

We use a 10Base2 Ethernet transceiver for the physical layer of ICEDnet. It may seem that this is an antiquated standard to follow, but it has pedagogical advantages. Students can directly probe the ICEDnet and see actual data collisions, that is, an actual Carrier Sense Multiple Access and Collision Detect protocol in action. While this is not that common in today's LANs, with switching and

point-to-point connections becoming the norm, the basic concept of resource contention in a communication medium is still very important, such as in wireless systems.

ICED sys-card

The sys-card is the equivalent of a motherboard for the ICED computer except that the CPU is remotely located in the host. While locating the CPU away from its memory is normally not done, in this case it let us use a basic commercial, off-the-shelf FPGA board (the EVC1) with its associated software and host interface support. It also helps emphasize to students the implications of a slow memory system.

Figure 4 shows the ICED sys-card. It has the following items and connected components:

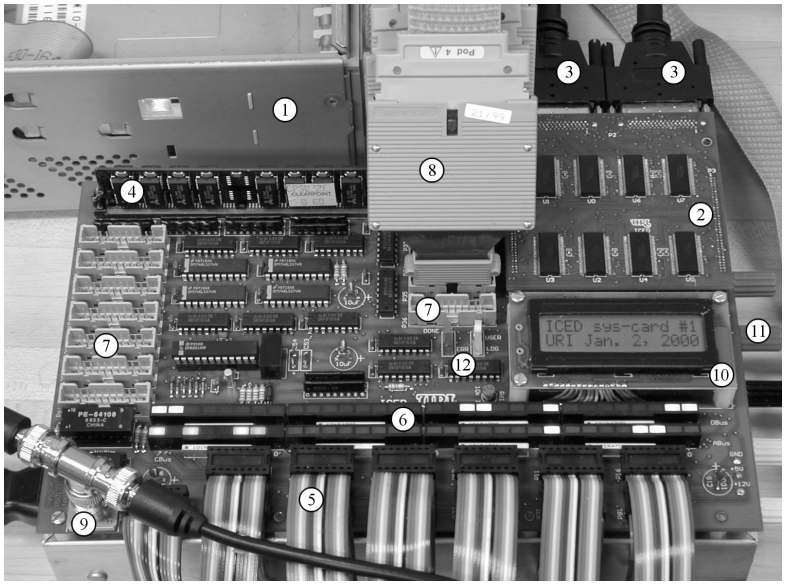

Figure 4. ICED sys-card. *(Photo by Laurette Bradley.)*

1. host computer;
2. d-card;
3. Ibus connectors (two groups of 32-bidirectional signals each) from host computer;
4. ICED computer dynamic RAM—8 Mbyte standard SIMM;
5. sys-card to protoboard connectors—thirteen 16-pin DIP connectors;
6. Ibus signal indicator LEDs—32 (top) for data, 24 (lower-right) for address, and 8 (lower-left) for control;
7. logic analyzer quick connectors—thirteen 16-signal connectors;
8. logic analyzer digital signal probe—16 signals;
9. ICEDnet connector—10Base2 physical layer standard;
10. LCD display—16 characters by 2 rows, general purpose;
11. DC power outputs—+5, +12, –5, –12 V (the protoboard currently only uses +5); and
12. EVC1 status display—four LEDs.

While this hardware resides on the sys-card, we move all the data and control connections from the sys-card to the protoboard. Several DIP connectors communicate with the students' circuits on the protoboard. All d-card signals (after the transceivers) go to the DIP connectors, as do all signals from the SIMM and the transceiver configuration signals. The sys-card and d-card are URI-designed, custom printed-circuit boards. The d-cards were assembled externally, the sys-cards at URI. All testing is performed at URI.

Protoboard and interface logic

The protoboard is a generic circuit-prototyping board composed of several push-in connection arrays. The DIP connectors from the sys-card plug into one or more of these arrays. There is one protoboard per student working group, kept for the project's duration, which holds the interface and memory control circuits and also acts as a patchboard. The protoboard is only attached to the sys-card during a laboratory session.

Logic analyzer

Each ICED lab station has its own high-speed HP 1662CS logic analyzer with 64 data channels and an integrated two-channel digitizing oscilloscope. Students can view and control each analyzer from its companion Sun host via the department's LAN. Viewing takes place via an X-Window system. Students also use the LAN to download disassembly information to the analyzer, as well as to connect the analyzer to a software analyzer running on the host.

Comments

The ICED Protosys laboratory station offers students a great deal of flexibility and exposes them to hardware and software interaction.

ICED students learn to use modern industrial EDA software tools.

Once we install all software tools, students will have access to many interesting capabilities with the lab equipment. In particular, students will be able to correlate an analog view of a signal with a digital view of the signal, with its particular machine and assembly instruction, and finally with the corresponding original high-level source instruction of the test program. This will let students see the effect on all levels of a design change at one of these levels, and thereby see the effects of hardware and software tradeoffs in an extremely concrete way.

ICED software tools and aids

Our students use several key software packages for the purposes of hardware design, hardware programming, and for signal and program variable visualization.

EDA tools

ICED students learn to use modern industrial EDA software tools. These tools are necessary for complex, digital designs, which require the EDA tools' automation, simulation, and verification capabilities before fabrication.

We use commercial software donated by Mentor Graphics and Xilinx. Students follow a typical design flow through the tools[7] and use them throughout the curriculum, dividing learning time over many courses. It is not essential or helpful to use multiple competing vendors' tools since once students learn a given tool, learning a different vendor's version of the same tool is relatively easy. The situation is similar to programmers learning a new language, where one imperative language is about the same as any other, conceptually. By using one set of tools, we standardize the department on the set and students do not waste time learning extra detailed information.

Other software

ICED also exposes students to software design and debugging aids. In particular, students use a disassembler and a symbol utility within the logic analyzer to interpret and annotate the prototype's bus operation. Individual assembly instructions on the bus can then be recognized by the logic analyzer and displayed to the student in both binary and mnemonic forms.

Uht wrote a retargetable assembler called icasm to accommodate students' different instruction sets. One version of icasm supports the canned CPU instruction set (IcpuED). Students can use icasm directly or to process the output of their compiler.

ICED students write a complete compiler for a simple high-level language for their prototype computer. Students use special Hewlett-Packard software running on the host and information generated by their compiler during an application's compilation run to show the correspondence between high-level source code and machine instructions on the host and the logic analyzer.

ICED operation and challenges

The ICED lab stations are expensive and we only have five. However, students can design on any of the Department of Electrical and Computer Engineering or the College of Engineering's Sun workstations or Dell Windows NT PCs. Students only need to use a full lab station at a lab's end for physical debugging and cost and performance investigations.

There are problems with a multiterm, multiyear project: students may get out of sequence due to leaves of absence, part-time study, student exchange program participation, and so on; a student's partner may drop out; or a group may fall far behind. To help in these situations, students may access a canned design to restart at the beginning of each core course, should they desire or need a fresh start. For example, a predesigned CPU is available at the beginning of the senior-year compiler design course.

There is a danger in having canned designs available, such as weary students prematurely giving up on their own designs, where the use of their own designs is one of our key goals. We are still struggling with this issue. One possible solution is some sort of long-term grade benefit for students who use their own designs.

Another ongoing issue is creating a bal-

anced distribution of the overall project among the core courses, especially those of computer architecture and computer design. Including some material on detailed computer and logic design early on (a bottom-up approach) helps clarify the principles of instruction set design (a top-down approach) and firmly grounds students in reality. However, too much low-level design work early on can leave insufficient room in the computer architecture course for traditional course topics We address the latter issue in the following years by expanding the three-credit architecture course into a three-credit lecture-only course and a two-credit lab course.

We've noticed a disturbing trend in our students' approach to hardware design. In the past, making a design error could be close to fatal for a student's project, since much of the hardware was built with handwired small- and medium-scale ICs. (Of course, this is still true for custom chip design.) Therefore students discovered, correctly, that they needed to spend a lot of time thinking about and checking their designs before putting hand to wire. More recently, simulation helped to verify the design. But now the hardware itself is programmable, and if students make a mistake, it's easy to change. They can hack in a fix, reroute, and reconfigure. This ease of correction creates a problem. Students now tend to approach their hardware design tasks like many approach software design, by sitting down at a terminal and rapidly beginning a design with little thought given to the task at hand.

Although these bad designs do not waste hardware, they certainly do waste students' time as well as create bad and buggy designs. It is an area of concern since we don't want hardware to start crashing at the same rate as many software packages. Fortunately, consumers seem to have little tolerance for buggy hardware, and the problem may correct itself in the marketplace.

We instruct students that the first step in hardware design is to draw designs and waveforms with paper and pencil and to continue analyzing the design until they're sure it's right. Only after this step should they use the EDA tool and start entering the design. Some students listen; more do the second time.

The ICED curriculum is a work in progress. To date, we have designed and built all of the custom hardware, and its testing is underway. Some tasks remain in setting up the laboratory software. We are performing an evaluation process of the curriculum that so far has had encouraging results.[7] Our evaluations will continue, and we will continue to publish them as results become available.

The URI ICED approach is one solution toward solving competing demands on computer engineering curricula. The ICED curriculum is a novel attempt to bring the problems and opportunities of a long-term project to undergraduates. MICRO

Acknowledgments

ICED is supported in part by the National Science Foundation through grant DUE-9751215, by a grant from the Champlin Foundations of Rhode Island, and with matching funds from the Office of the Provost at URI. Mentor Graphics, Xilinx, and Virtual Computer Corporation provide software donations. Virtual Computer Corporation, American Power Conversion Corporation, and Clearpoint Memory provide hardware donations.

We are indebted to Carlo Tognina for his dedicated and excellent work on the ICED hardware.

References

1. See selected papers of Augustus K. Uht; http://www.ele.uri.edu/~uht.
2. A.K. Uht, "The URI Integrated Computer Engineering Design (ICED) Curriculum: Progress Report," to be published in *Proc. American Society of Engineering Educators Annual Conference*, ASEE, Washington, D.C., 2000.
3. *Technical Committee on Computer Architecture Newsletter: Special Issue on Computer Architecture Education*, D.R. Kaeli, ed., IEEE Computer Society, Los Alamitos, Calif., June 1996.
4. J.O. Hamblen et al., "Using Rapid Prototyping in Computer Architecture Design Laboratories," *Technical Committee on Computer Architecture Newsletter: Special Issue on Computer Architecture Education*, IEEE Computer Society, pp. 44-52, June 1996.
5. *Technical Committee on Computer Architecture Newsletter: Special Issue on*

Computer Architecture Education, D.R. Kaeli, ed., IEEE Computer Society, Sept. 1997.
6. G.M. Brown and N. Vrana, "A Computer Architecture Laboratory Course Using Programmable Logic," *IEEE Trans. Education*, Vol. 38, No. 2, May 1995, pp. 118-125.
7. A.K. Uht and Y. Sun, "The Laboratory Environment of the URI Integrated Computer Engineering Design (ICED) Curriculum," *Proc. Frontiers in Education Conf.*, ASEE, Washington, D.C., 1998, pp. 331-336.

Augustus K. Uht is a licensed professional engineer and an associate professor at the Department of Electrical and Computer Engineering at URI. He is interested in processor architecture and education. Uht received a BS and a MEng(Elect) from Cornell University and a PhD from Carnegie Mellon University. He is affiliated with the IEEE; ACM; the National Society of Professional Engineers (NSPE); HKN, the electrical engineering honor society; and Sigma Xi, the science honorary society.

Jien-Chung Lo is a professor at the Department of Electrical and Computer Engineering at URI. His research interests include I_{DDQ} testing and reliable computing. Lo received an MS and a PhD from the Center for Advanced Computing Studies at the University of Louisiana, Lafayette. He is a member of the IEEE Computer Society.

Ying Sun is a professor at the Department of Electrical and Computer Engineering and is coordinator of biomedical engineering, both at URI. His research interests include medical instrumentation, medical imaging, and modeling of physiological systems. Sun received a PhD in electrical engineering from Worcester Polytechnic Institute. He is a member of the IEEE.

James Daly is a professor at the Department of Electrical and Computer Engineering at URI. He is interested in integrated circuit design and education. Daly received a BS from the University of Connecticut, and an MEE and a PhD from Rensselaer Polytechnic Institute. He is affiliated with the IEEE, HKN, and Sigma Xi.

James Kowalski is an associate professor and chair of the Computer Science Department at URI. His interests include neural networks, genetic algorithms, and data mining. Kowalski received a BS in physics, an MA in philosophy, and a PhD in mathematical logic from the University of Notre Dame. He is affiliated with the IEEE Computer Society, the American Association for Artificial Intelligence, and the ACM.

Direct questions about this article to Augustus K. Uht at the University of Rhode Island, A-219 Kelley Hall, 4 East Alumni Ave., Kingston, RI 02881-0805; uht@ele.uri.edu.