CPLD-Based Design in the Introductory Computer Engineering Course

David J. Ahlgren
Trinity College

INTRODUCTION

During the last three years the emphasis in ENGR221L--Digital Circuits and Systems, the introductory computer engineering course at Trinity College, has shifted toward the increased use of VHDL and complex programmable logic devices (CPLD's). This evolution has encouraged team projects that employ top-down design and concurrent engineering approaches. Working student designs included a single-chip stepper motor controller, a stimulus-response timer, finite state machines, and several tiny microprocessors. Completed in four weeks, including two or three formal lab sessions, fully operational four-bit microprocessors were designed by student teams each year from 1994-96. These processors each had a sixteen-member instruction set adequate for writing short, but instructive, programs. For example, programs that performed elementary operations on arrays were developed on the 1995 design (CPU221/95), which incorporated immediate, direct, and indexed auto-increment addressing modes.

These successes have shown that students become productive quickly when introduced to a well-integrated tool-set in the first course. Students are able to complete design projects that are not feasible in the available time when attempted using standard chips, and the emphasis in the first course can be shifted away from wiring and troubleshooting toward system-level design.

This paper describes the course and laboratory, presents design projects undertaken in Fall, 1996, and reflects on the results.

INTRODUCTORY COMPUTER ENGINEERING COURSE

The goal of the first course in computer engineering at Trinity College, ENGR221L--Digital Circuits And Systems, is to develop a working knowledge of digital devices and systems in the context of computer-aided design. The course covers number systems and codes; Boolean Algebra and combinational logic; flip-flops, registers and counters; finite state machines; memory devices and systems; programmable logic; and processor and interface organization. The lecture presents VHDL descriptions of familiar logic components in parallel with standard descriptions. These components include basic logic gates (AND, NAND, etc.); familiar MSI devices (decoders, multiplexer, adders, etc.); flip-flops and registers; and finite state machines. These examples introduce students to VHDL structure and syntax, provide useful models that
can be altered and applied when solving design problems, and develop implicitly the knowledge of a useful VHDL subset. This subset consists of the following VHDL statements and constructs: ENTITY, ARCHITECTURE, SIGNAL, TYPE, PORT, PROCESS, IF-THEN-ELSE, CASE..WHEN, Conditional Assignment (WHEN...ELSE structure), and 'EVENT.

The ENGR221L laboratory, which meets each week for three hours, emphasizes CAD-based design using standard components (primarily LSTTL) and programmable logic devices. In the last three years the ENGR221L laboratory has employed Altera CPLD's and the associated CAD tool-set, Max+Plus II. Since most of the students are sophomores, ENGR221L offers a first exposure to engineering laboratory practice. Many plan to major in engineering, but some will major in such fields as computer science, psychology, and chemistry. Students generally work in teams of two in the lab, but on large projects several teams will combine their efforts. Teams develop their designs before coming in into the laboratory, producing a pre-lab document that includes schematic diagrams and simulation results.

In 1996 four introductory experiments exposed students to basic logic gates, the application of Boolean Algebra and Karnaugh Maps, use of the B^2 Logic CAD package (available from Beige Bag Software, info@beigebag.com), and breadboarding and troubleshooting techniques. The experiments were as follows: (Week 1) Breadboarding and Basic Logic Gates--an introduction to the breadboarding system, logic gates (AND, OR, NOT, etc.), simulation, and troubleshooting; (Week 2) Digital Circuits for the Automobile--the development of two designs--a traffic light controller for a four-way intersection, and a safety device that counts and displays the number of doors open in a hatchback; (Week 3) Communication System with Parity--the design of a four-bit transceiver including parity generator and detector (odd and even, with ability to deliberately induce errors). Units are connected to form a network; (Weeks 4 & 5) Adder/Subtractor--the design of a circuit that adds or subtracts four-bit numbers. Inputs are a control bit, which selects the operation, and two four-bit two's complement numbers. The results are displayed on a seven-segment display that includes a plus/minus sign. Student solutions to this problem typically employ 10-12 LSTTL packages to implement full adder, complementer, and seven-segment decoder and driver circuitry.

In 1996, the sixth and seventh weeks introduced students to Altera's Max+Plus II CAD tools including the schematic (graphic) editor, compiler, waveform editor (for generating test waveforms), simulator, text editor, and hierarchy display tool. During these sessions, students re-designed their four-bit adder/subtractor and display in two ways: 1) by using LSTTL cells from the Max+Plus II library; and 2) by developing a VHDL description. Each design was programmed into an Altera EPM5032 device and tested on the breadboard. These exercises formed a smooth transition from LSTTL-based design to CPLD-based design while demonstrating the productivity advantages of the CPLD implementations. The consequence is that in the future students think first of CPLD-based solutions when approaching design problems.

After completing a one-week exercise that introduced standard LSTTL flip-flops, registers, and counters, students completed the "Mini-ALU" project, which sharpened skills with Max+Plus II and VHDL and illustrated the operation of a simple ALU. The handout presented VHDL code for a two-bit ALU. The exercise required students to compile this code, create test vectors using
the waveform editor, and carry out a full simulation. Students increased the ALU word size from two to four bits, added a second ALU register (B), implemented the instructions A PLUS B and A AND B, and added a status register containing zero, carry, and negative flags. This exercise offered a head start to teams planning to design a "tiny" CPU as a final design project.

FINAL DESIGN PROJECTS

In 1996, teams were allowed to pick a project from a provided list, define their own project, or join with two other teams in designing a 4-bit microprocessor, the CPU221/96. The students were allowed to employ an LSTTL-based approach, but they were encouraged to use the Altera tools, VHDL, and CPLD's. A one-page statement of intent was due on November 22, and students demonstrated their projects and handed in their final reports on Wednesday, December 18.

The instructor suggested the following problems: 1) vending machine that dispenses 35¢ and 60¢ items, accepts nickels, dimes, and quarters, and makes change; 2) digital stopwatch, controlled by start and stop buttons, that displays times from 0.0 to 99.9 seconds; 3) pulse-width modulator for d.c. motor speed control; 4) taillight control system for a new automobile; 5) digital stimulus/response timer with display in milliseconds; 6) energy efficient washroom for campgrounds; 7) ultraviolet flame detector interface used on fire-fighting mobile robot; 8) elementary calculator with keyboard encoder and seven-segment display; and 9) CPU221/96 processor. All teams chose a project from this list; the first team (2 students) picked the digital stopwatch and the second team (3 students) chose the stimulus/response timer. Six teams (12 students) split into two design groups (A and B) to work on CPU designs.

The digital stopwatch and the reaction timer problems required students to consider the development of accurate time-base circuitry. The stopwatch team used a phase-locked loop to generate a 600 Hz signal locked to the 60 Hz line. This frequency was divided by six to generate a 100 Hz standard yielding 0.01 sec. resolution. Requiring approximately 1080 usable gates, all stopwatch circuitry except for the PLL fit into an Altera EPM7064 84-pin CPLD. The reaction timer team used a 1 MHz quartz crystal oscillator as a frequency standard, and derived an on-chip 10 KHz clock that yielded 0.1 ms. resolution. Except for the quartz oscillator, all reaction timer components fit into an Altera EPM7096. Developed using Altera LSTTL library cells, this design required six 74LS160 decade counters, one 74LS76 J-K flip-flop, one AND gate, and four 74LS47 decoder/drivers and it required 1020 equivalent gates on the CPLD. Both the stopwatch and the reaction timer worked perfectly in simulation, and the programmed chips worked the first time.

The 1996 tiny processor designs evolved from projects successfully completed in 1994 and 1995. The assignment handout was cryptic, including only a processor block diagram, an initial instruction set (which students were encouraged to modify), a register transfer planning matrix, and sample programs written using the initial instruction set. This material was explained in class since this processor was used as an example to illustrate the stored program principle.

The CPU consisted of the ALU, control unit (CU, containing finite-state machine and control logic), program control and addressing block (PCAB), and bi-directional data bus interface unit.
The BIU was provided to the students. The initial design described in the handout utilized 128 memory locations—requiring a 7-bit address bus—half of which were dedicated to program ROM, and half to SRAM. (The physical system used the lowest 64 bytes of a standard EEPROM chip and 64 bytes of a type 2114 4-bit SRAM). Arithmetic operations assumed two's complement data. The CPU was driven by a variable frequency clock; a second clock phase was derived on the chip.

The CPU221/96 design included the following registers:

- **A, B**: 4-bit ALU registers
- **IR**: 4-bit instruction register
- **X**: 7-bit index register
- **PC**: 7-bit program counter
- **MAR**: 7-bit memory address register
- **SR**: status register (containing Zero, Negative, Overflow flags)
- **TMP**: 8-bit temporary register used, for example, to store PC and X-register offsets.

The initial instruction set consisted of sixteen operations including loads and stores, a universal logic operation (NOR), left and right shifts, an "update the index register instruction" that adds an offset to X, a conditional branch, and an unconditional branch (Table I). Limited by the number of opcode bits, some memory reference instructions could not use every addressing mode; for example, the NOR instruction used immediate mode addressing only.
TABLE I: CPU221/96 Initial Instruction Set

<table>
<thead>
<tr>
<th>Operation</th>
<th>Mnemonic</th>
<th>Opcode</th>
</tr>
</thead>
<tbody>
<tr>
<td>Load A immediate</td>
<td>LDAI</td>
<td>0000</td>
</tr>
<tr>
<td>Load A direct</td>
<td>LDAD</td>
<td>0001</td>
</tr>
<tr>
<td>Load A indexed</td>
<td>LDAX</td>
<td>0010</td>
</tr>
<tr>
<td>Add immediate</td>
<td>ADDI</td>
<td>0011</td>
</tr>
<tr>
<td>Add indexed</td>
<td>ADDX</td>
<td>0100</td>
</tr>
<tr>
<td>NOR immediate</td>
<td>NORI</td>
<td>0101</td>
</tr>
<tr>
<td>Store A direct</td>
<td>STAD</td>
<td>0110</td>
</tr>
<tr>
<td>Store A indexed</td>
<td>STAX</td>
<td>0111</td>
</tr>
<tr>
<td>Logical shift left</td>
<td>LSL</td>
<td>1000</td>
</tr>
<tr>
<td>Logical shift right</td>
<td>LSR</td>
<td>1001</td>
</tr>
<tr>
<td>Load X immediate</td>
<td>LDXI</td>
<td>1010</td>
</tr>
<tr>
<td>Load X direct</td>
<td>LDXD</td>
<td>1011</td>
</tr>
<tr>
<td>Store X direct</td>
<td>STXD</td>
<td>1100</td>
</tr>
<tr>
<td>Update X immediate</td>
<td>UDXI</td>
<td>1101</td>
</tr>
<tr>
<td>Branch unconditional</td>
<td>BRA</td>
<td>1110</td>
</tr>
<tr>
<td>Branch if zero</td>
<td>BEQ</td>
<td>1111</td>
</tr>
</tbody>
</table>

The key to efficient logic design for this CPU was completion of the planning matrix. The matrix lists the instructions on one axis and the machine states on the other. The number of states needed to execute an instruction varies; some instructions are executed in as few as four states, others require as many as ten. For each instruction, students wrote down in the matrix state-by-state a sequence of register transfers that causes execution of the instruction in one cycle. The control logic outputs that enable these transfers were also entered. Having completed this matrix, students were quickly able to write a Boolean expression for each control logic output. Completion of the planning matrix was the key step in the overall design process.

This project required students to manage their groups efficiently and to divide the work so that teams could work concurrently. Each team was responsible for developing VHDL code for a CPU sub-system (ALU, CU, PCAB). Teams discovered quickly the need to describe the signal interfaces among the sub-systems. Once each team's VHDL-based module was fully simulated and verified, students used the Max+Plus II graphic editor to connect the sub-systems in a top-level schematic. This top-level design was compiled into a targeted CPLD and simulated fully. Most of the 1994-96 ENGR221L CPU designs were tested further by creating simulated ROMS containing virtual programs, connecting these ROMS to the processor, and executing those programs using the Max+Plus II simulator.

RESULTS

The first group of six tackling the CPU221/96 project (Group A) consisted of six engineering majors or prospective engineering majors. Group A organized themselves efficiently and completed the planning matrix without delay. They modified the instruction set by replacing the left- and right-shift instructions by a NOT instruction and a NAND immediate instruction. All sub-systems were developed in VHDL and each was simulated and determined to be correct.
Encountering problems with simulating the bi-directional data bus, Group A was not able to verify the top level design. Nevertheless, the design was programmed into an Altera EPM9320 device which was tested on the breadboard. These tests indicated that some sub-systems, including the finite state machine, operated correctly but that the memory addressing system did not. Breadboarding problems may have slowed Group A.

Group B formed an interdisciplinary corporation, the BRACCK Associates (Brian, Rumana, Antony, Chris, Chris, and Kris), consisting of two prospective engineering majors, three computer science majors, and one psychology major. This team analyzed planned their solution carefully before writing VHDL code. They eliminated the shift instructions, replaced the immediate-mode NORI by an indexed-mode NORX, and added branches on overflow and negative conditions. In addition, Group B doubled the memory size, increasing the address bus from seven to eight bits. Group B was able to simulate the top-level design, and they executed programs stored in a simulated ROM. The completed CPU was programmed into an 84-pin Altera EPM9320 CPLD, requiring 220 logic cells and approximately 4100 usable gates. Group B designed their pinout to match the wiring on a breadboard that was used successfully by the CPU221/95 design group. LED’s on this breadboard monitored the contents of the CPU registers and the data bus. A variable frequency clock allowed monitoring of program flow. The first program run on Group B’s CPU221/96 hardware, a bubblesort that sorted a list of five numbers stored in SRAM, ran correctly. (Execution of the bubblesort was videotaped by the group and will be shown at the ASEE conference.)

REFLECTIONS AND ASSESSMENT

The incorporation of powerful, state-of the art CAD tools has allowed students to tackle complex design projects in the first course; the CPU design projects at Trinity College provide firm evidence. A subset of VHDL has been introduced by presenting examples describing common hardware elements. There is no need to delay the introduction of these tools until a higher-level course. Using them, students applied team-based concurrent engineering methods in the first course, worked in a hierarchical design environment, and considered solutions from an architectural viewpoint. The simulation tools have raised the probability of producing working projects.

The knowledge of CPLD synthesis gained in ENGR221L has supported later projects, independent studies, and senior design projects. For example, students are planning to develop a VLSI-based processor, an expanded version of the CPU221/96 design, that will be sent to MOSIS for fabrication. Several students are involved in the Spring, 1997 semester in designing CPLD-based designs used in mobile robotics; these include a d.c. motor controller based on a PLL, a stepper motor control chip, and an ultraviolet flame sensor interface. Supported by a NASA grant, another student is developing a DSP processor based on Altera CPLD's.

Altera devices were chosen for this work for a number of reasons. The EEPROM- and EPROM-based components in Altera's EPM5000, EPM7000, and EPM9000 series are re-programmable and do not require external components for initialization or re-configuration as FPGA's would. Thus it is straightforward to apply and re-use these devices in the laboratory. Students have become productive quickly using the Altera Max+Plus II CAD package; Max+Plus II provides
powerful and well-integrated tools, a deep set of features that can be learned in a "just in time" fashion, a library of components that eases the transition from TTL-based design to CPLD-based design, and the ability to produce fast prototypes. The Altera Tools for Higher Education Program has donated the CPLD’s that Trinity students have used as well as a copy of Max+Plus II for each PC and HP/UNIX workstation in the computer engineering lab. (In 1994 and 1995, we successfully used the PC-based tools on a Windows 3.1-based network with a Windows NT server. Network reliability and overall productivity were higher with the HP/UNIX workstations used in 1996.) Altera has sent regular software updates without charge.

The consequences of this shift in emphasis in the first course are significant. The projects create intense student interest in digital design, and the sense of accomplishment among students is great. The successful tiny processor design projects especially have helped to identify ENGR221L as a course that offers opportunities to design real systems using state-of-the-art technology.

ACKNOWLEDGEMENTS

The author thanks the Altera Tools for Higher Education Program for its generous and sustained support. The author also appreciates the dedicated and skillful work of his teaching assistants Michael Guillorn '98 and Brian Hall '98.

DAVID J. AHLGREN
Professor of Engineering
Trinity College, Hartford CT 06106
dahlgren@trincoll.edu

DAVID J. AHLGREN
Professor of Engineering
Trinity College, Hartford CT 06106
dahlgren@trincoll.edu