From Zero to One: An Introduction to Digital Design and Computer Architecture

David Money Harris, *Senior Member, IEEE*, and Sarah Harris, *Member, IEEE*

***Abstract*— All Harvey Mudd College Engineering students take a one-semester course on Digital Design and Computer Architecture. The lectures are closely accompanied by a series of laboratory projects leading to the implementation and programming of a simplified MIPS microprocessor on an FPGA. We are developing a textbook to help other instructors interested in such a unified course. Our experience shows that hardware-software co-design for reconfigurable computing is a perfectly natural theme for an introductory digital course.**

***Index Terms*—digital design, computer architecture, FPGA, education**

I. Introduction

ALL Harvey Mudd College Engineering majors are required to take a one-semester course called Introduction to Computer Engineering (E85). The course covers digital design and computer architecture. It culminates with a project in which students build a MIPS processor on an FPGA and program the processor to control an external speech synthesizer chip.

E85 has been organized around tightly integrated lectures and FPGA-based labs since 1999 [1]. In Fall 2005, the course incorporated Hardware Description Languages (HDL) and moved to a modern FPGA platform with an I/O device lab. We are presently writing a textbook to support the class.

This paper describes the guiding principles and structure of the E85 course. It discusses the lab assignments and the new textbook. It concludes with assessment results and discussion of some of the challenges in structuring such a course.

II. Guiding Principles

We believe that the first course in digital design can be one of the most exciting subjects studied by EE, CS, and CE majors. The material is relatively easy to understand, requiring very little mathematics beyond zero and one. The subject is also conducive to hands-on laboratory projects in which students build interesting digital systems with a modest amount of effort compared to other engineering disciplines. The projects are appealing to the large fraction of students who were drawn to the field because they liked to build things and who may have been frustrated with the largely theoretical courses at the beginning of many curricula. And the results are spectacular; students leave the course with tangible skills to build systems that would have seemed impossible for most before they entered the class.

We teach a unified course covering both digital design and computer architecture. In our experience, the two topics are highly synergistic. Students are fascinated to learn how a microprocessor works and recognize their new knowledge as a major milestone in their education. The microprocessor is a canonical digital system, complex enough to illustrate all of the issues in basic digital design, yet simple enough for a first course.

We believe that the laboratory component should be an inseparable part of the introductory course because students learn best by doing. In the 1980's and 1990's, laboratory exercises using 74-series TTL logic were well suited to such a course because the TTL chips concretely support the notion of logic gates and because they are easy to breadboard. Now TTL systems are largely obsolete and the vast majority of modern digital design targets Field Programmable Gate Arrays (FPGAs) or Application-Specific Integrated Circuits (ASICs). FPGAs and ASICs use very similar design methods and FPGAs are ideal for rapid prototyping, so FPGAs have become the natural target for digital design labs. The key pedagogical challenge is that FPGA design is more abstract, especially when HDLs are employed.

Similarly, we have found students benefit from learning to design at the schematic level before transitioning to HDL because they need to think of hardware as distinct from software. Once they have experience with schematics, we introduce HDL as a shortcut for implying the hardware, constantly revisiting the link between hardware and the HDL for describing it. Students are eager to learn HDL because they understand that it is how real designs are done today.

Putting these principles together, we find that hardware-software co-design for reconfigurable computing is a perfectly natural theme for an introductory course in digital systems.

Manuscript received November 27, 2005.

David Money Harris and Sarah Harris are with the Department of Engineering, Harvey Mudd College, Claremont, CA 91711 {David_Harris, Sarah_Harris}@hmc.edu.

III. Course Structure

E85, along with introductory courses in mechanical, chemical, materials, electrical, and systems engineering, is required of all general Engineering majors at Harvey Mudd College. It is offered in the fall and spring semesters to a total of 80-90 students per year, including some CS majors and students from adjacent Claremont colleges. The prerequisite is a freshman-level introduction to Java programming taught by the CS department. E85 is primarily taken by sophomores and juniors, most of whom are taking four or five other classes at the same time. Hence, the course must pack a large amount of content into a modest amount of the students' time.

The course is usually taught with two 75-minute lectures each week for fifteen weeks. The syllabus is given in Table 1. Students turn in a problem set and a laboratory assignment every week except during exams and holidays. The laboratory exercises are a key hands-on element of the class and will be discussed in detail in the next section. Problem sets reinforce concepts that would be too time-intensive to explore in lab; they also help prepare students for the midterm and final exams.

IV. Lab Assignments

Table 2 lists the E85 laboratory assignments. These laboratories are completed using Xilinx ISE 6.3i with ModelSim 6.0 for simulation and Synplify Pro 8.1 for synthesis.

In the first three labs, the students use schematic entry to understand basic logic circuits. They use combinational logic gates in the first two labs followed by sequential logic circuits in the third lab. They follow the steps of (1) describing the problem, (2) writing truth tables, (3) deriving the Boolean equations, and (4) translating the equations to hardware.

In the fourth and fifth labs students transition to a hardware description language (HDL). In our class, we use Verilog, but

TABLE I
E85 Syllabus

No	Date	Topic	Assignment
1	31-Aug	Introduction: digital abstraction; binary numbers; bits and bytes; logic gates	
2	5-Sep	Transistor-level implementation; truth tables, Boolean expressions	
3	7-Sep	Boolean algebra; K-maps	PS 1
4	12-Sep	X's and Z's; timing, hazards	Lab 1
5	14-Sep	Sequential circuits: SR latches, D latches, flip-flops, clocking	PS 2
6	19-Sep	Finite State Machines	Lab 2
7	21-Sep	Dynamic discipline; metastability	PS 3
8	26-Sep	Introduction to Hardware Description Languages (HDLs): Verilog	Lab 3
9	28-Sep	More Verilog	PS 4
10	3-Oct	Building Blocks I: mux, decoder, priority encoder, counter, comparator	Lab 4
11	5-Oct	Building Blocks II: Arrays: RAMs, ROMs, PLAs, FPGAs	PS 5
12	10-Oct	Number systems: fixed & floating point, unsigned & signed	
	12-Oct	Midterm: take-home	
	17-Oct	Fall Break: no class	
13	19-Oct	Arithmetic: addition & subtraction, multiplication	PS 6
14	24-Oct	MIPS instruction set and registers	Lab 5
15	26-Oct	Branches & procedure calls	PS 7
16	31-Oct	Addressing modes	Lab 6
17	2-Oct	Linking & launching applications	PS 8
18	7-Nov	Single-cycle processor datapath	Lab 7
19	9-Nov	Single-cycle processor control	PS 9
20	14-Nov	Multicycle processor	Lab 8
21	16-Nov	Exceptions	PS 10
22	21-Nov	Pipelining	Lab 9
23	23-Nov	Pipelining hazards and stalls	PS 11
24	28-Nov	Memory-mapped I/O	Lab 10
25	30-Nov	Memory hierarchy, latency & throughput Caches	PS 12
26	5-Dec	Memory system optimization, Virtual memory	Lab 11
27	7-Dec	Advanced architecture: a sampler	
	14-Dec	Final: take-home	

the text has side-by-side coverage of Verilog and VHDL. The hardware and software are also compatible with both HDLs. In the fourth lab, students also design a finite state machine, as in lab 3, but this time by describing it using an HDL. The parallels between schematic entry and HDL code are made clear because labs 3 and 4 show both methods for approaching a similar problem. The students view the schematic synthesized from the HDL code using the synthesis tool, Synplify Pro. They can then correct any errors if the synthesized schematic is not what they expected. The students are prepared to write effective, synthesizable HDL code because of the hardware basis they developed in the first three labs.

In labs 6 and 7, the students learn how to approach digital systems from a software perspective by writing MIPS assembly code. They learn about the architecture of the processor, including its register and memory conventions, by simulating their code on the SPIM simulator.

In labs 8 through 10, the students link the domains of software and digital hardware by designing and simulating the MIPS processor executing test code. Lab 8 gently guides students through the design of a complex system, while labs 9 and 10 turn them loose to do their own design.

In lab 8, the students are given the structure of the single-cycle processor and fill in the necessary logic. To test their processor, the students translate a MIPS assembly code into machine code, load it into the memory used by the processor, and simulate the program running on their completed single-cycle processor using ModelSim. This lab is their first experience building a complex digital system, and the HDL structure provided exemplifies good design practice for future labs.

In labs 9 and 10 the students are much more on their own to work as independent circuit designers. Given well-defined interfaces between the processor and memory, and, within the processor, between the datapath and control, they implement the controller and datapath. As in lab 8, their completed processor is tested by loading a machine code program into the processor's memory and running it on the processor in simulation using ModelSim.

The final lab illustrates hardware/software co-design for a real-world application. In this lab, students interface their multicycle MIPS processor, implemented on the Spartan 3 FPGA, with a speech synthesizer chip using memory-mapped I/O. The students control the speech chip by developing assembly code to drive the memory-mapped control and data signals.

In Lab 4, students downloaded their designs onto a custom Spartan 3 FPGA board over the Xilinx Parallel

TABLE II
E85 LABORATORY ASSIGNMENTS

Lab	Description	Design Method
1	1-bit Full Adder	Schematic
2	Control Decoder for MIPS	
3	Adventure Game Finite State Machine	
4	Turn Signals Finite State Machine	HDL
5	32-bit ALU	
6	Computing Fibonacci Numbers	Assembly
7	Software Floating Point Addition	Language
8	MIPS Single-Cycle Processor	HDL
9	MIPS Multicycle Processor Control	
10	MIPS Multicycle Processor	
11	Processor, I/O Interface: Speech Synthesis	HDL & Assembly

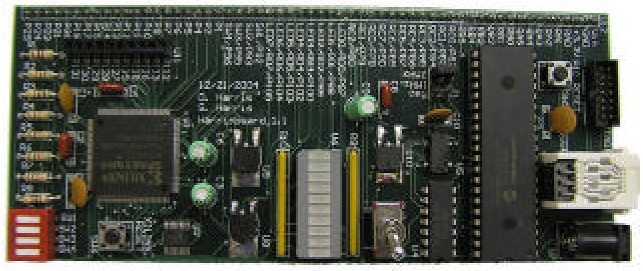


Fig. 1. Spartan 3 FPGA board.

Cable IV so they could watch their FSM output on LEDs. Likewise, Lab 11 is downloaded to a board with a prewired speech chip and speaker. The board, adapted from an upper division elective, is shown in Fig. 1 and described further in [2]. The labs progressively build upon each other. The full adder from Lab 1 is used in the ALU in Lab 5, which in turn is used in the processors in Labs 8 through 11. The control decoder from Lab 2 is also used in the processor in Lab 8.

Students can access the Engineering Computational Facility 24 hours a day to work on their labs, or may install the CAD software on their personal computers. Undergraduate lab assistants hold scheduled lab hours each week, help struggling students with troubleshooting, and grade the labs. Typically there is one lab assistant for every 25 students and the assistant is present for 2-4 hours each week.

In summary, by using configware as the basis for learning digital design and computer architecture, the students retain the value of a hands-on design experience without the tedium of breadboarding discrete gates. They learn modern digital design practices, computer architecture, and software/hardware co-design through a series of labs that progressively build upon principles learned in previous labs.

TABLE III
TEXTBOOK CONTENTS

1 From Zero to One

- 1.1 The Game Plan
- 1.2 The Art of Managing Complexity
- 1.3 The Digital Abstraction
- 1.4 Number Systems
- 1.5 Logic Gates
- 1.6 Logic Levels
- 1.7 CMOS Transistors
- 1.8 Summary and A Look Ahead

2 Combinational Logic Design

- 2.1 Introduction
- 2.2 Boolean Equations
- 2.3 Boolean Algebra
- 2.4 From Logic to Gates
- 2.5 Multilevel Combinational Logic
- 2.6 X's and Z's, Oh My
- 2.7 Karnaugh Maps
- 2.8 Multiplexers
- 2.9 Timing
- 2.10 Summary

3 Sequential Logic Design

- 3.1 Introduction
- 3.2 Latches and Flip-Flops
- 3.3 Synchronous Logic Design
- 3.4 Finite State Machines
- 3.5 Timing of Sequential Logic
- 3.6 Parallelism and Pipelining
- 3.7 Summary

4 Hardware Description Languages

- 4.1 Introduction
- 4.2 Combinational Logic
- 4.3 Structural Modeling
- 4.4 Sequential Logic
- 4.5 More Combinational Logic
- 4.6 Finite State Machines
- 4.7 * Parameterized Modules
- 4.8 Test Benches
- 4.9 Summary

5 Digital Building Blocks

- 5.1 Introduction
- 5.2 Combinational Building Blocks
- 5.3 Sequential Building Blocks
- 5.4 Arithmetic
- 5.5 Arrays
- 5.6 *Chip Implementation Options
- 5.7 *System Implementation Options
- 5.8 Summary

6 Architecture

- 6.1 Introduction
- 6.2 Assembly Language
- 6.3 Machine Language
- 6.4 Programming
- 6.5 Addressing Modes
- 6.6 Assembling and Loading
- 6.7 * Other MIPS Instructions
- 6.8 Summary

7 Microarchitecture

- 7.1 Introduction
- 7.2 Performance Analysis
- 7.3 Single-Cycle Processor
- 7.4 Multicycle Processor
- 7.5 Pipelined Processor
- 7.6 * HDL Representations
- 7.7 * Exceptions
- 7.8 Summary

8 Memory Systems

- 8.1 Introduction
- 8.2 Caches
- 8.3 Virtual Memory
- 8.4 Memory Mapped I/O
- 8.5 Performance Analysis
- 8.6 Summary

9 Advanced Microarchitecture

- 9.1 Introduction
- 9.2 Branch Prediction
- 9.3 Superscalar Pipelines
- 9.4 Out of Order Execution
- 9.5 Summary

Appendix: C Programming

V. TEXTBOOK

Until this year, E85 historically used Patterson & Hennessy's *Computer Organization and Design* text published by Morgan Kaufmann [3]. The textbook elegantly covers assembly language programming, computer architecture, and microarchitecture using the MIPS processor. However, it assumes a previous course in digital logic. With the generous encouragement of Patterson, Hennessy, and Morgan Kaufmann, we are writing a new textbook for a unified course building on the strengths of Patterson & Hennessy's MIPS designs.

Table 3 lists the contents of the new book. Optional sections are indicated with a *. As of February 2007, Chapters 1-8 have been written and are being reviewed. Morgan Kaufmann plans a January 2007 publication date. The book targets one semester or two-quarter introductory courses in electrical engineering, computer science, and computer engineering.

Another unique element of the textbook is its coverage of HDLs. We cover Verilog and VHDL in chapter 4, after combinational and sequential logic design, but before the design of larger components or microarchitecture. We teach HDLs not as a programming language (as far too many books do), but as a shorthand for specifying digital hardware. The chapter is organized around various digital hardware structures (e.g. combinational logic, registers, FSMs, and hierarchy). Both Verilog and VHDL describing each structure are presented in side-by-side format. Below the code is a schematic synthesized from the HDL to drive home the relationship between code and hardware. Instructors can select either HDL, and students can easily learn the other once they have mastered the first. The HDL is then emphasized throughout subsequent chapters.

Chapter 7, discussing MIPS microarchitecture, ties together all of the previous chapters. It features the designs of single-cycle, multicycle, and pipelined processors shown in Figure 2 that are closely adapted from Patterson & Hennessy. The chapter illustrates combinational and sequential circuit design and the use of digital building blocks. It draws on the assembly language programming material. And it features a HDL model of the single-cycle processor illustrating the design of a complex system.

VI. Assessment

E85 has been taught as a unified course on digital design and computer architecture for the past seven years by five different instructors. It generally receives teaching evaluations above 6.0 on a 7 point scale. Instructors in the upper division digital design electives find that E85 students are well prepared for advanced work. Recent student comments include:

I enjoyed how the labs built upon each other, it was cool seeing it all come together.

The labs were really fun.

Students did complain about fighting with bugs in Xilinx Version 2 and 4, but have been more satisfied with Version 6.3i.

To make sure the workload stays under control, students are asked to report the time they spend on class assignments. In the past three semesters, the workload has averaged about 4 hours per lab and 2.5 hours per problem set, for a total of 6.5 hours per week. This is typical for a 3-unit course at HMC. Moving from Xilinx ISE version 4 to 6 saved about half an hour per week on average because the tool became more stable.

The labs were labor-intensive to develop, but reasonably inexpensive to maintain. Many undergraduates have contributed to the lab development, enhancing their own education as well. The department hires one undergraduate lab assistant for every 15-20 students; the assistant generally works about 2 hours per week. Xilinx ISE is free to academic institutions and ModelSim and Synplify Pro have nominal academic pricing. The FPGA boards cost under $100; Digilent also sells a Spartan 3 board for $99 [4] and Xilinx has been generous about donations. The board is connected to a PC in the department general-purpose computer lab; dedicated lab facilities are under consideration.

We have found that the schematic viewer in Synplify Pro is essential to teaching HDL in a first course; in previous years when no viewer was available, students had no way to understand what good synthesizable coding style meant and HDLs were too abstract.

VII. Future Enhancements

In the past, one FPGA board has been shared across the entire class, so only Labs 4 and 11 use actual hardware. In the future, we would like to add more FPGA stations so that students can physically implement Labs 1 and 2 on the FPGA as well.

The progression of labs worked well when all labs used schematics, but is problematic because the Xilinx schematic editor does not play gracefully with Synplify Pro Verilog synthesis. As a result, Labs 2 and 5 need to be rethought.

We are currently upgrading to Xilinx ISE 8.1, which may eliminate the need for the third-party ModelSim simulator.

VIII. Conclusions

Harvey Mudd has offered a very successful required introductory course on digital design and computer architecture using FPGAs with schematic entry and HDL. The lab assignments are closely coupled to the lectures. We believe that students benefit greatly from the hands-on design experience and find it highly relevant because it uses modern technology. We also find that teaching schematic-level design before HDL is important for students to avoid the trap of viewing HDL as a programming language. The course gives students a solid foundation for future work in both hardware and software design for reconfigurable computing systems. We are developing a textbook to support the integration of digital design and computer architecture in a first course.

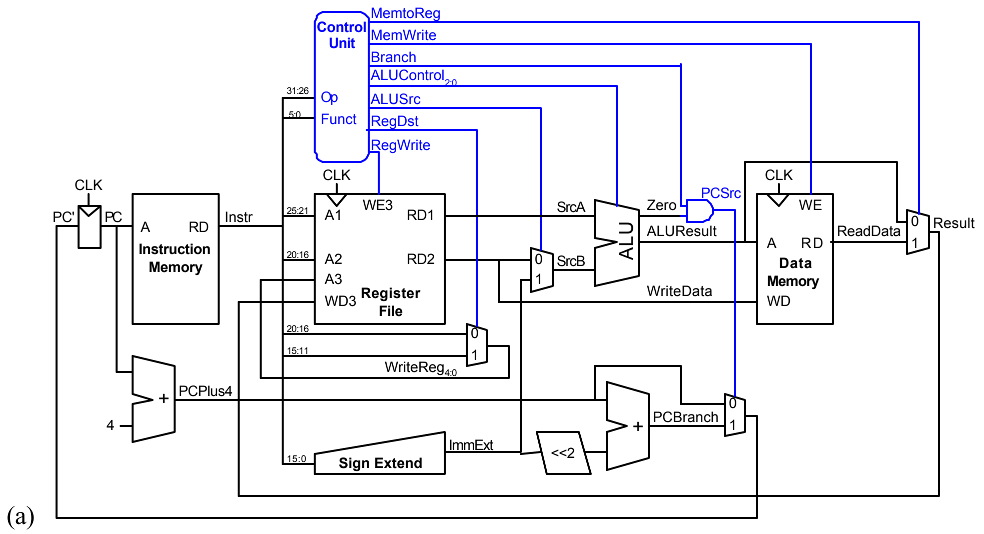
References

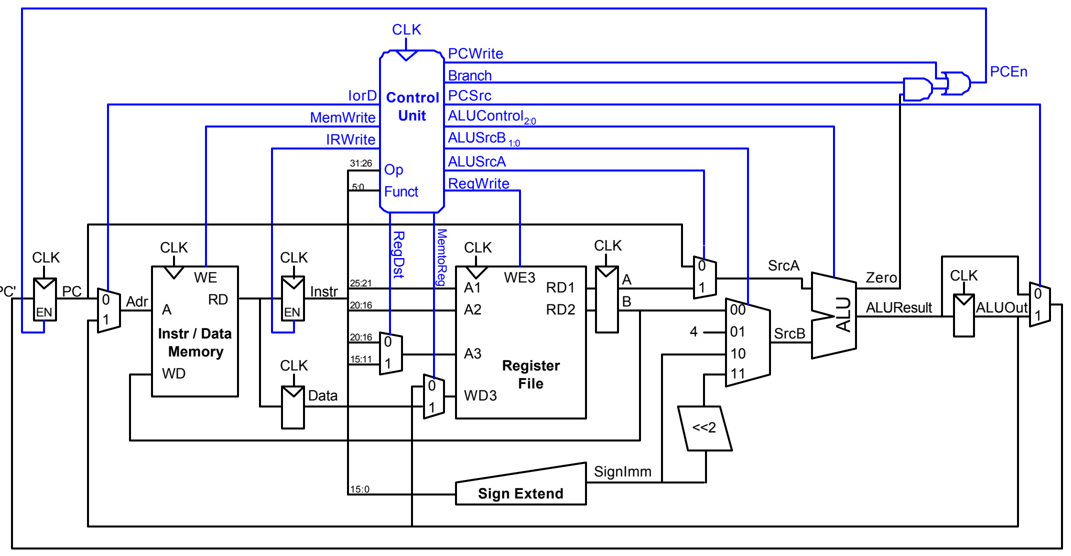
[1] D. Harris, “A Case for Project-Based Design Education,” *Intl. J. Engineering Education*, vol. 17, no 4-5, pp. 367-369, 2001.

[2] S. Harris and D. Harris, “Inexpensive Student-Assembled FPGA / Microcontroller Board,” *Microelectronics Systems Education Conf.*, June 2005, pp. 101-102

[3] D. Patterson and J. Hennessy, *Computer Organization and Design*, 3rd Ed., San Francisco: Morgan Kaufmann, 2005,

[4] www.digilent.com


(a)


(b)

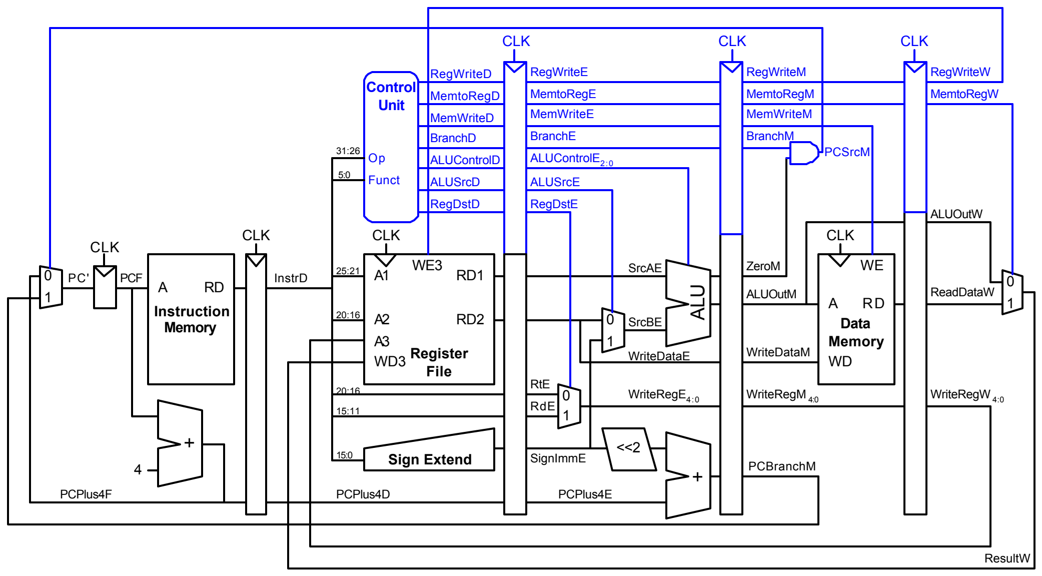

(c)

Fig. 2. (a) Single-cycle, (b) Multicycle, and (c) Pipelined MIPS processors.