Digital Design and RISC-V Computer Architecture Textbook

Sarah L. Harris†
Electrical and Computer Eng.
University of Nevada
Las Vegas, NV USA
Sarah.Harris@unlv.edu

David Harris
Department of Engineering
Harvey Mudd College
Claremont, CA USA
David_Harris@hmc.edu

ABSTRACT

This paper describes the authors' *Digital Design and Computer Architecture: RISC-V Edition* textbook. The book presents a unified 1- or 2-semester course on digital design and computer architecture. We have found that learning these topics together clarifies and solidifies understanding of both concepts. The textbook begins by describing digital design concepts and techniques, from number systems, logic gates, and transistor-level gate design to synchronous sequential circuits such as finite state machines and other common digital building blocks. It then builds on these concepts to teach computer architecture and processor design by introducing the RISC-V instruction set architecture (ISA), showing how to design three RISC-V processors with limited instructions, and describing various memory organizations, including caches and virtual memory. The textbook also describes logic design using hardware description languages (HDLs), covering SystemVerilog and VHDL side-by-side. The optional appendices and online chapters introduce the C programming language, embedded system design, and practical aspects of digital design including breadboarding, ASIC design, and transmission lines.

CCS CONCEPTS

• Architectures • Embedded Systems • Logic

KEYWORDS

Digital Design; Computer Architecture; RISC-V; Education

1 Introduction

The RISC-V computer architecture is the first widely adopted open-source computer architecture. Its open-source nature makes it particularly accessible, and its relevance to both current and emerging processors and systems is increasingly important. We have written a new textbook, *Digital Design and Computer Architecture: RISC-V Edition*, © Elsevier, 2021 [1], to teach logic design principles and then build on that foundation to teach the RISC-V computer architecture and processor design.

We have taught digital design and computer architecture as an integrated single-semester or multi-semester sequence for over 15 years using our prior textbooks, which also cover digital design and computer architecture but focus on the MIPS [2] and ARM® [3] architectures. We have found that connecting the dots between these two topics provides deeper understanding of both concepts, enables hands-on learning, and empowers the students to build on both concepts to design more complex digital circuits and to use computer architecture principles in applications including software algorithms and embedded system design [4, 5].

This approach has been effective and popular world-wide. For example, in the U.S. in 2020 alone, prior editions of our book, which focus on the MIPS and ARM® architectures, were used by 10,000 university students. These prior editions are available in seven languages (English, Chinese, Japanese, Korean, Spanish, Russian, and Portuguese), and we expect the RISC-V Edition will also be translated into other languages.

2 Textbook Content

This new textbook describes digital design techniques and building blocks in the first half of the textbook and then shows how to use these building blocks to design a RISC-V processor and memory system in the second half of the textbook. Table 1 shows the textbook's table of contents.

The book starts by describing digital design principles, beginning with binary numbers, addition, and logic gates (Chapter 1) and moving on to combinational and sequential logic design (Chapters 2 and 3) and then common building blocks such as adders, multipliers, and memories (Chapter 5). Chapter 4 gives an introduction to hardware description languages, including both SystemVerilog and VHDL. However, this chapter may be skipped without sacrificing understanding of the concepts in the other chapters.

The second part of the book focuses on computer architecture by first introducing the RISC-V instruction set architecture (ISA) in Chapter 6, including describing the RISC-V instruction set and registers, showing how to convert high-level code to assembly, and describing machine encodings.

Workshop on Computer Architecture Education'21, June, 2021, Online

Table 1: Textbook Table of Contents

The book also discusses RISC-V extensions, including compressed instructions (the RVC extension), and floating-point instructions (RVF/D). The textbook then shows how to build three versions of a limited-instruction RISC-V processor: a single-cycle processor, multicycle processor, and pipelined processor (Chapter 7). The book also describes advanced microarchitecture techniques including branch prediction, out-of-order execution, and multiprocessors. The textbook concludes by discussing memory organization, including caches and virtual memory, in Chapter 8.

Optional online chapters and appendices cover embedded I/O (Chapter 9), practical digital design techniques (Appendix A), and an introduction to the C programming language (Appendix C). Appendix B gives a summary of RISC-V instructions, and it is available on the inside covers of the textbook. The C appendix may be taught before discussing the RISC-V instruction set architecture for students who have no prior programming experience.

Topics discussed in the appendix on practical digital design considerations include chip packages, logic families, breadboard circuit assembly, datasheets, transmission lines, and engineering economics. The optional embedded I/O chapter describes common microcontroller interfaces and techniques, including general-purpose I/O, timers, interrupts, serial interfaces such as SPI and I^2C, and other common peripherals such as LCDs and VGAs. Chapter 9 uses SparkFun's Red-V RedBoard or Thing Plus board, which include SiFive's FE310-G002 RISC-V system-on-chip (SoC), to demonstrate how to program and use peripherals using a RISC-V processor.

Table 2: Example 1-Semester Course Syllabus

Lecture	Topics	Readings	Assignment Due
0	Intro: digital abstraction, numbers	1.1-1.5	
1	Logic gates, static discipline, transistors	1.6-1.9, A1-A7	
2	Combinational logic design	2.1-2.8	Problem Set 1
3	Timing, sequential circuits	2.9-2.10, 3.1-3.2	Lab 1: Comb Logic
4	Finite state machines	3.3-3.4	Problem Set 2
5	Dynamic discipline, metastability	3.5-3.7	Lab 2: Comb Logic
6	Hardware description languages: SystemVerilog, part I	4.1-4.3	Problem Set 3
7	SystemVerilog, part II	4.4-4.10	Lab 3: FSM from Gates
8	Arithmetic circuits	5.1-5.2	Problem Set 4
9	Fixed and floating-point number systems	5.3	Lab 4: FSM in SV
10	Sequential building blocks, arrays	5.4-5.7	Problem Set 5
11	Midterm Review		Lab 5: Building Blocks
	Midterm		
12	C Programming, part I	C1-C.7	
13	C Programming, part II	C.8-C.11	
14	Assembly language	6.1-6.3.5	PS 6
15	Function calls in assembly	6.3.6	Lab 6 C Programming
16	Machine language	6.4-6.9	PS 7
17	Single-cycle processor datapath	7.1-7.3.1	Lab 7 C with Peripherals
18	Single-cycle processor control, SystemVerilog	7.3, 7.6	PS 8
19	Multicycle processor	7.4	Lab 8 Single-Cycle Proc.
20	Pipelining	7.5.1-7.5.2	PS 9
21	Advanced architecture: a sampler	7.7	Lab 9 Multicycle Datapath
22	Introduction to memory systems, caches	7.7	PS 10
23	Virtual memory	7.7	Lab 10 Multicycle Control

2 Course Structure

The material can be taught as a 1-semester or 2-semester course, with the first semester focusing on digital design and the second semester on computer architecture. Table 2 shows an example syllabus for the 1-semester course. The course is set up so that the material is introduced with increasing levels of depth. We first introduce the concepts during lecture and work through example problems with the class. Then, the students complete weekly problem sets to learn the materials in a low-overhead, exploratory manner. Finally, students complete weekly hands-on labs to learn the material in-depth and to be able to use the material to solve real-world problems.

3 Companion Material

The textbook's companion material includes exercise solutions, laboratory exercises and solutions, lecture slides, textbook figures, example syllabi and exams, and source code from the textbook, including all HDL examples. All materials are in their source format – i.e., Microsoft Word (.docx) format for lab instructions and solutions, PowerPoint (.pptx) format for lecture slides, Visio (.vsdx) format for figures, and text format for code. This enables instructors to readily use and adapt the material as desired.

The instructor version includes solutions to all exercises and labs, whereas the student version includes only odd numbered exercise solutions and lab instructions, but not solutions. The instructor companion materials also include the HDL (both SystemVerilog and VHDL) for all RISC-V processors (single-cycle, multicycle, and pipelined) introduced in the textbook. These processors implement a reduced set of RISC-V instructions: `add`, `sub`, `and`, `or`, `slt`, `addi`, `andi`, `ori`, `slti`, `beq`, and `jal`.

4 Laboratory Assignments

Laboratory assignments consist of ten hands-on exercises ranging from designing digital circuits using schematics or SystemVerilog to programming in C and building and testing

RISC-V processors in SystemVerilog on an FPGA board. Table 3 lists the labs included as part of the companion materials for the textbook.

The labs use the hardware and software listed in Table 4. All of the software is free and the labs may be completed in simulation only, although we have found that it is a richer experience when the students are able to view their designs working in hardware. However, because the software used is free and the hardware is optional, the labs may be completed without cost, as needed.

The digital design labs (Labs 1-5) and the processor design labs (Labs 8-10) use Intel's Quartus Lite or Web Edition software (formerly from Altera) for design entry and ModelSim, either Starter or Intel FPGA Edition, for simulation. After simulation, the designs can be optionally programmed onto the DE2-115 board that contains Intel's Cyclone IVE field programmable gate array (FPGA). In 2021, the DE2-115 board costs $600, with a reduced academic price of $309. The labs could be readily adapted to other FPGA boards and programming tools, such as Digilent's Nexys A7 or Basys FPGA board and Xilinx's Vivado design suite.

The C programming labs (Labs 6 and 7), use PlatformIO, a free embedded systems IDE (integrated development environment) that is an extension of Visual Studio Code (VSCode), to load and run programs on SparkFun's RED-V RedBoard ($40) or Thing Plus board ($30), which both include SiFive's FE310-G002 RISC-V SoC.

Table 3: Labs

No.	Description
1	Schematic Design: 1-bit Full Adder
2	Schematic Design: 7-Segment Display
3	Schematic Design: FSM
4	SystemVerilog: FSM
5	SystemVerilog: 32-bit ALU & Testbench
6	C Programming: Matrix Multiplication
7	C Programming: Simon Says Game - LEDs & Switches
8	SystemVerilog: Single-Cycle Processor
9	SystemVerilog: Multicycle Datapath
10	SystemVerilog: Multicycle Processor

Lab 1 guides students to design, build, simulate, and test a 1-bit full adder using schematic entry in Intel's Quartus software. Then, students use ModelSim to simulate their design and Quartus to program the FPGA DE2-115 board with their design. They then test their design in hardware using the switches and LEDs on the DE2-115 board. Lab 2 is similar to Lab 1 – but this time the students design a more complex circuit, a 7-segment display decoder.

Lab 3 shows how to design a finite state machine (FSM) using schematic entry. Starting with Lab 4, where students design another FSM, students use SystemVerilog to implement their digital circuits. Lab 5 guides students in building an ALU and writing a testbench to debug and test their design.

Labs 5 and 6 transition to writing C programs for the RED-V RedBoard or Thing Plus board. Lab 5 uses software only to guide students in writing linear algebra and array manipulation functions. Lab 6 shows how to use the general-purpose I/O pins on the RED-V boards to play a Simon Says game with LEDs and switches that must be wired up to the board. Labs 6 and 7 are optional and may be skipped if boards are not available.

Table 4: Required Software & Hardware

Software	Link
Quartus Lite/Web Edition	https://fpgasoftware.intel.com
ModelSim Intel FPGA Edition	https://fpgasoftware.intel.com/?product=modelsim_ae#tabs-2
Visual Studio Code (VS Code)	https://code.visualstudio.com/download
PlatformIO	Extension within VS Code
Hardware	**Link**
DE2-115 Board	http://de2-115.terasic.com
RED-V RedBoard or Thing Plus Board	https://www.sparkfun.com/products/15594

In Lab 8, students are given the SystemVerilog code for the single-cycle processor discussed in the book and extend it to support additional RISC-V instructions. Students also use the ALU they built in Lab 5 and must also modify the test program, written in RISC-V assembly and translated to machine code, and the testbench to simulate and test their extended processor. In Labs 9 and 10, students use the building blocks (register files, memories, multiplexers, ALUs, registers, etc.) provided in Lab 8 to implement the multicycle processor introduced in the book.

The book's companion website also provides optional labs including RISC-V assembly labs and additional C programming labs that interface with an accelerometer and a buzzer to create a digital level and pulse-width modulated sound and light.

Instructors may readily extend the existing labs to other applications or designs. For example, the logic design labs (1-

5) can be extended to other digital systems by swapping out the target circuit. For example, instead of building a 7-segment display decoder, students could build a priority circuit or prefix adder. Further, students could create and implement their own FSM designs after completing Lab 3 or 4. The C programming labs (Labs 6 and 7) could also be extended to other software algorithms or peripherals.

5 MOOC

In addition to the textbook, we have also developed two massive open online courses (MOOCs) to accompany the textbook, which are available through EdX. The book content is divided into two courses, one on digital design and one on computer architecture. These courses may be found by searching for "Digital Design HarveyMuddX" [6] and "Computer Architecture HarveyMuddX" [7] on EdX. These courses are currently instructor-led, but we plan on creating self-paced versions. The videos are freely available upon registration at EdX, but the graded exercises and certificate cost a fee.

6 Architecture Comparisons

Like most architectures, the RISC-V architecture includes both similarities and differences when compared to other contemporary RISC-V architectures, such as the MIPS and ARM® architectures.

RISC-V shares many commonalities with the MIPS architecture, including similar machine instruction formats (i.e., I-type, R-type, and J-type), instruction mnemonics (i.e., `lw`, `sw`, `addi`, `slt`, etc.), and register naming (i.e., `s0`, `t0`, `a0`, `sp`, etc.). However, RISC-V gets rid of MIPS idiosyncrasies, such as the branch delay slot, branching relative to PC+4 (instead of PC), and inconsistent register locations in machine instruction encodings. RISC-V also opts for more complex immediate encodings to optimize hardware. Finally, RISC-V and is growing in its commercial use while MIPS is declining.

When compared with ARM®, RISC-V also has a small set of instruction formats and includes 16-bit (compressed) instructions, which can be compared to ARM's 16-bit Thumb instruction set. These smaller instructions minimize program size and, thus, memory usage, which is often critical in embedded and low-power systems. But unlike ARM®, RISC-V does not include conditional execution or complex indexing modes. By not including these, RISC-V minimizes hardware size and complexity. Lastly, ARM® requires licensing, which can be prohibitive in terms of cost and design time, whereas RISC-V is open source.

7 Conclusions

We have found that teaching digital design and computer architecture together enhances the clarity and understanding of both topics. We have written a textbook that starts from the basics of binary numbers and logic gates and then guides students in designing increasingly complex digital circuits, which culminates in their learning about the RISC-V architecture and then designing, building, and testing a RISC-V processor.

The RISC-V architecture is a particularly relevant and well-suited architecture to study, especially for students learning computer architecture for the first time, because it is open-source, offers an extensible instruction set to support a broad range of processors, and has a growing commercial market – while, at the same time, the architecture is straightforward and thus enables student understanding. By building up from basic principles and building blocks, students understand the RISC-V architecture and processor design from top to bottom.

ACKNOWLEDGMENTS

We would like to acknowledge Josh Brake, Assistant Professor at Harvey Mudd college, for his contributions to the MOOC and Chapter 9 of the textbook. We also thank our many reviewers and the team at Morgan Kaufmann for their support in publishing this textbook.

REFERENCES

[1] S. Harris and D. Harris, *Digital Design and Computer Architecture, RISC-V Edition*, Morgan Kaufmann, 2021
[2] D. Harris and S. Harris, *Digital Design and Computer Architecture, 2md Edition*, Morgan Kaufmann, 2011
[3] S. Harris and D. Harris, *Digital Design and Computer Architecture: ARM® Edition*, Morgan Kaufmann, 2015
[4] D. Harris and S. Harris, *From Zero to One: An Introduction to Digital Design and Computer Architecture*, the First International Workshop on Reconfigurable Computing Education, March 1, 2006, Karlsruhe, Germany
[5] S. Harris and D. Harris, *ARM-Based Digital Design and Computer Architecture Curriculum*, 17th International Conference on Information Technology–New Generations (ITNG 2020), April 5-8, 2020 (virtual)
[6] *Digital Design*, HarveyMuddX, edx.org
[7] *Computer Architecture*, HarveyMuddX, edx.org