RVfpga: Computer Architecture Course and MOOC Using a RISC-V SoC Targeted to an FPGA and Simulation

Dr. Sarah L. Harris, University of Nevada, Las Vegas

Dr. Harris is a Professor at the University of Nevada, Las Vegas (UNLV) in the Electrical & Computer Engineering Department. She earned her M.S. and Ph.D. at Stanford University and has worked at Hewlett Packard, Nvidia, and the Technical University of Darmstadt. Before joining the UNLV faculty in 2014, she was a faculty member at Harvey Mudd College for ten years. Her research interests include embedded systems, biomedical engineering, and robotics, and she has co-authored three popular textbooks, most recently Digital Design and Computer Architecture: RISC-V Edition in 2021.

Daniel Chaver Martinez, University Complutense of Madrid, Spain
Luis Piñuel
Olof Kindgren
Robert C.W. Owen

RVfpga: Computer Architecture Course and MOOC using a RISC-V SoC Targeted to an FPGA and Simulation

Sarah L. Harris[1], Daniel Chaver[2], Luis Piñuel[2], Olof Kindgren[3], Robert Owen[4]

[1]University of Nevada, Las Vegas, [2]University Complutense of Madrid, [3]Qamcom Research & Technology, [4]Imagination Technologies

Abstract

RISC-V FPGA, also written RVfpga, is a freely available course that provides instructions and resources, including the unobfuscated RISC-V system-on-chip (SoC) itself, to show how to readily use and understand a RISC-V SoC, from writing C and assembly programs down to understanding and expanding the system. These materials bridge the gap between the availability of the open and royalty-free RISC-V instruction set architecture (ISA) and actually being able to use and experiment with a commercial RISC-V processor/SoC and the RISC-V toolchain. In addition to providing the SoC source code in Verilog/SystemVerilog and showing how to readily use the RISC-V toolchain to compile, debug, and load C and RISC-V assembly programs onto the SoC, both in simulation and on an FPGA (field programmable gate array), RVfpga shows how to expand the system to add peripherals and how to explore and modify the microarchitecture, including adding instructions, measuring performance using built-in performance counters, and exploring microarchitectural features, from the most fundamental aspects, such as pipelining and caches, to other more specific and advanced capabilities, such as superscalar execution, non-blocking loads and divide operations, secondary ALUs for resolving data hazards, unaligned loads and stores, scratch pad memories for both instruction and data, and advanced branch prediction. We also show how to use several simulators: the Whisper instruction set simulator (ISS) and three Verilator-based simulators: RVfpga-Trace, RVfpga-ViDBo, and RVfpga-Pipeline. The simulators enable users to fully use the materials without the expense of purchasing an FPGA board; thus, the course may be completed without cost. This paper also describes an RVfpga EdX MOOC (massive open online course) that we are completing, which includes ten in-depth chapters, accompanying videos and tutorials, and exercises. This online course can be used on its own or as a guide to instructors in how to present and teach the RVfpga content. In addition to these materials, we have also run ten 1-day RVfpga workshops that were taught world-wide over the past year. The RVfpga materials are most typically implemented as a two-semester course: the first being a junior-level course in digital design, computer architecture, and embedded systems with a follow-on course, at the senior/master's level, in microarchitecture, but the materials may also be used in a condensed 1-semester course or for self-study.

Keywords

RISC-V, FPGA, microarchitecture, architecture, embedded systems, SoC

1. Introduction

The recent and ongoing development of the open and royalty-free RISC-V architecture has become increasingly pervasive in industry and research [1]. Because it is open and royalty free, it is more accessible for both academia and industry, and the hurdle of costly and time-consuming licensing is removed. Other features that make RISC-V appealing are its extensibility, which enables a wide range of microprocessors, from low-cost microcontrollers to high-performance cores. Although the architecture is open and royalty-free, most RISC-V processor cores are not. However, several companies and groups have provided commercial open-source cores, including the VeeR cores from CHIPS Alliance, which were originally SweRV cores from Western Digital [2]. The ecosystem built around the RISC-V architecture, including the toolchain and simulators, has also been steadily developing since RISC-V's inception in 2010.

Despite these developments, the ability to readily access, use, and work with the RISC-V architecture and open-source RISC-V systems still faces barriers, including lack of understanding the architecture and its extensions, the inability to access, understand, and use RISC-V tools, and the difficulty in gathering and using all of the pieces needed to deploy, understand, and expand a commercial, non-trivial, RISC-V core and SoC with its accompanying development environment and tools. While some courses exist to address some or parts of these issues [3, 4, 5, 6, 7, 8], this RVfpga course aims to address all of these challenges by showing the complete path from system setup to using, programming, understanding, and expanding the RISC-V core and SoC, including learning how to use the RISC-V toolchain and simulators, how to use and understand a commercial RISC-V SoC targeted to an FPGA, and how to extend and modify the SoC and core [9]. In addition to developing the materials for this course, including a Getting Started Guide, twenty labs, hands-on tutorials, and accompanying exercises and solutions to many of the exercises, the course also provides the HDL (Verilog and SystemVerilog) source code for the RISC-V SoC. The materials are provided in their source format (.docx for labs and instructions, .pptx for slides, and .txt for text files), to enable instructors to readily adopt and adapt the materials for their own use. To increase accessibility, the authors have also given one-day workshops throughout 2022 and 2023 and are now completing an RVfpga MOOC (massive open online course) in EdX that will be available by summer 2023. After completing this course, users will walk away with a working RISC-V development environment and system that they know how to use and extend.

The course also embodies the collaboration inherent in open-source movements by being developed collaboratively by Imagination Technologies and their industry and academic partners, including the authors, RISC-V International, the Linux Foundation, and many other universities and industries. The course is freely available in both EdX and in its full download from the Imagination Technologies University Programme [10], which requires a short, several-minute registration [11].

The remainder of this paper describes the course goals as well as an overview of the course and its structure, including a brief introduction to the RVfpga core and SoC, a description of the required software tools (all of which are free) and optional hardware, and an overview of the twenty RVfpga labs. We then describe the RVfpga EdX course and the workshops that we have given internationally. We conclude by describing future improvements we plan to implement this year and by summarizing the course's features and what we have accomplished.

2. RVfpga Course Goals, Overview, and Structure

The RVfpga course aims to enable users to understand and use a commercial RISC-V core and system and then learn how to extend the system for learning, research, and experimentation. Users are expected to have a fundamental understanding of digital design and computer architecture in general before beginning the course. Such topics are covered in many textbooks, including *Digital Design and Computer Architecture: RISC-V Edition* [12]. The RVfpga course then builds on and expands those topics through hands-on labs and exercises.

The RVfpga course has two subsections, also referred to as subcourses, each of which could be implemented as a semester-long course. The first subsection focuses on learning to use the system, including programming the SoC, using existing peripherals, and extending the system to add other peripherals. The second subsection focuses on microarchitecture, including understanding the pipelined core and the system's microarchitectural features such as branch prediction, memory hierarchy, and hazard handling. Both subsections also show how to experiment with peripheral and microarchitectural modifications and extend the SoC. The EdX MOOC mainly covers the first subcourse, and the entire RVfpga course materials are available for free download from Imagination Technologies. This section describes the RVfpga system, the required and optional tools, and the twenty labs.

2.1 RVfpga System

The RVfpga system is an extended version of the open-source VeeRwolf SoC [13]. VeeRwolf (originally called SweRVolf) is based on the open-source VeeR EH1 core [2]. The source code for the RVfpga system is already provided with the course; however, we provide the links for the original repositories of the core and original SoC (VeeRwolf) in Table 1 for reference. The core and SoC are provided by CHIPS Alliance, an organization that develops and shares open-source hardware designs.

Table 1. Open-Source RISC-V Core and SoC

Core or SoC	Website
VeeR EH1 Core	https://github.com/chipsalliance/Cores-VeeR-EH1
VeeRwolf (SweRVolf) SoC	https://github.com/chipsalliance/Cores-SweRVolf

The VeeR EH1 core has a 2-way, 9-stage, in-order pipeline, as shown in Figure 1 [14]. It supports the RV32IMC RISC-V architecture; that is, the RV32I base instruction set with the multiply/divide (M) and compressed (C) extensions. The pipeline has two Fetch stages, Align and Decode stages, three Execute stages (integer execution, load/store path, or multiply path), and Commit and Writeback stages. The core also includes an out-of-pipeline divider that can take up to 34 cycles.

These stages are implemented using multiple hardware units within the core, including the instruction fetch unit (IFU) for the Fetch and Align stages, the Decode unit (DEC) for the Decode stage, the Execute Unit (EXU) for the integer execution stages (EX1-3) and multiply execution stages (M1-M3), and the Load Store Unit (LSU) for the load/store path (DC1-3), as shown in Figure 2. The core is expanded to become the VeeR EH1 core complex by including on-chip memories and peripheral interfaces, including closely-coupled memories for data and instructions (DCCM and ICCM), and AXI or AHB-Lite bus interfaces.

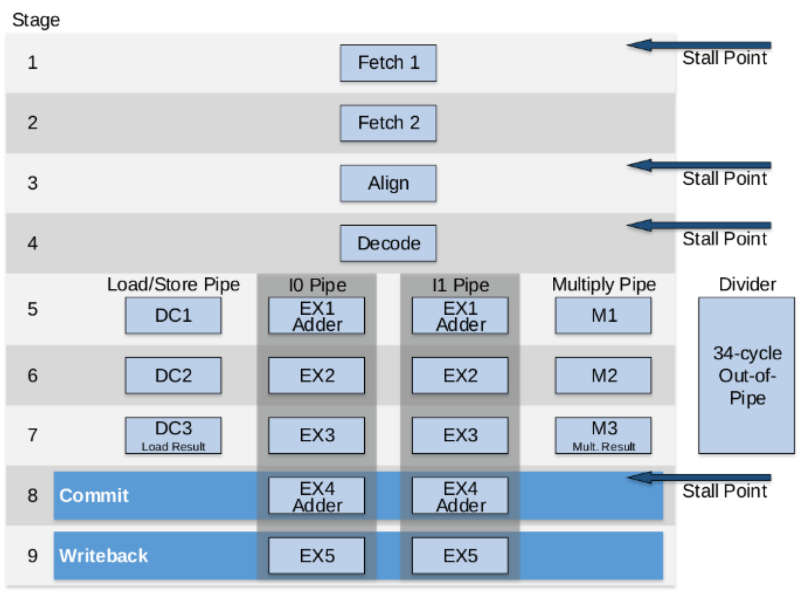

Figure 1. VeeR EH1 pipelined core (figure from [2])

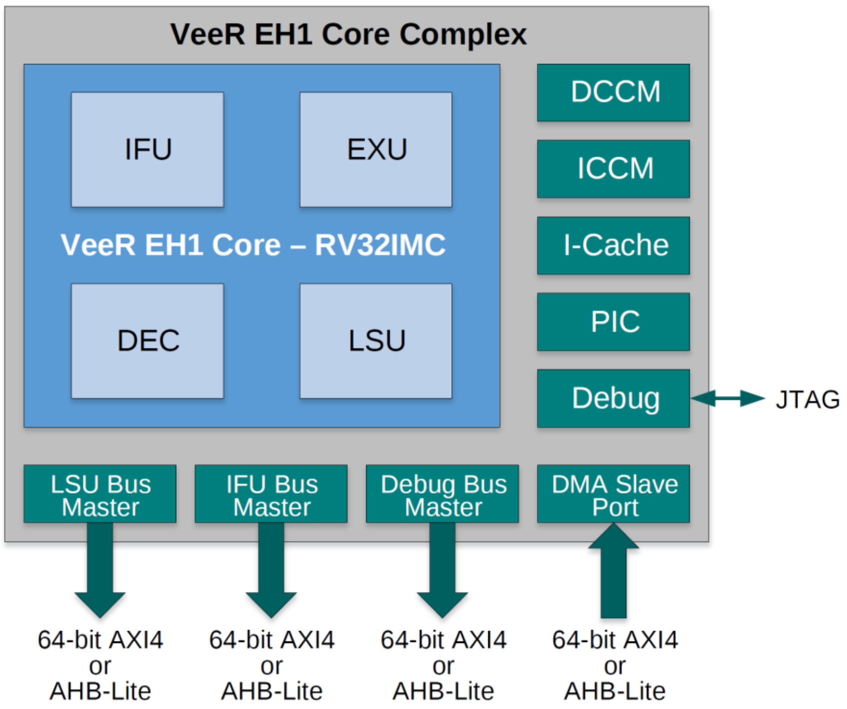

Figure 2. VeeR EH1 core complex (figure from [2])

The RVfpga SoC then connects the VeeR EH1 core complex to peripherals, memories, and interfaces, including GPIO (general-purpose I/O), 8-digit 7-Segment Displays (included in the System Controller), SPI, and UART interfaces as well as a boot ROM and a PTC (PWM/Timer/Counter), as shown in Figure 3. Table 2 shows the RVfpga memory map, which uses addresses from 0x80000000 - 0x80002FFF. As mentioned, the RVfpga SoC is an extended version of the VeeRwolf SoC; the interfaces in red in Figure 3 and that are starred (*) in Table 2 indicate interfaces that were added to the VeeRwolf SoC to create the RVfpga SoC. Although it is not explicitly shown in the figure, the 8-digit 7-Segment Displays were also added to VeeRwolf inside its System Controller (System-Ctrl) module.

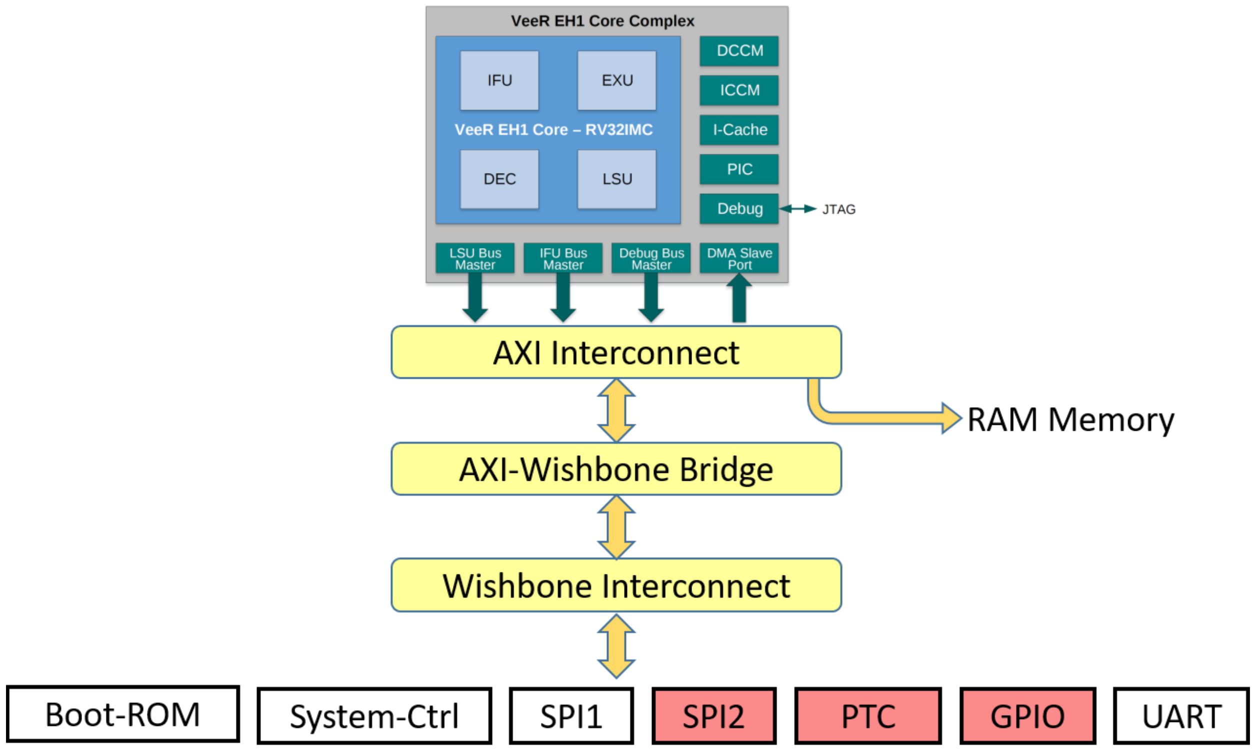

Figure 3. RVfpga SoC

Table 2. RVfpga Memory Map

System	Address
Boot ROM	0x80000000 - 0x80000FFF
System Controller	0x80001000 - 0x8000103F
SPI1	0x80001040 - 0x8000107F
SPI2*	0x80001100 - 0x8000113F
PTC*	0x80001200 - 0x8000123F
GPIO*	0x80001400 - 0x8000143F
UART	0x80002000 - 0x80002FFF

2.2 RVfpga Software, Tools, and Optional Hardware

The RVfpga course is supported in Linux, Windows, and macOS. All software required to use this course is free, as listed in Table 3. PlatformIO is a development environment that is an extension of Visual Studio Code (VSCode); it is the main tool that must be installed. Many of the remaining tools listed in Table 3 are installed automatically during the RVfpga setup process, particularly when installing PlatformIO, which is described in detail in the course. We provide links to all tools for the user’s convenience, although separate downloads are not needed for many of the tools beyond PlatformIO. The RISC-V toolchain includes compilers and debuggers and it is downloaded within the context of PlatformIO.

The remaining software tools listed in Table 3 are simulators [15]. The open-source Whisper instruction set simulator (ISS) was originally developed by Western Digital and is now available from CHIPS Alliance. Whisper ISS simulates RISC-V assembly code, independent of any hardware, so it offers functional simulation but not cycle-accurate simulation.

Table 3. Software Tools

Tool	Website
PlatformIO	https://platformio.org/ & https://code.visualstudio.com/Download
RISC-V Toolchain	https://github.com/riscv/riscv-gnu-toolchain
Whisper ISS	https://github.com/chipsalliance/VeeR-ISS
Verilator	https://github.com/verilator/verilator
GTKWave (used by RVfpga-Trace)	http://gtkwave.sourceforge.net/
RVfpga-ViDBo	https://github.com/artecs-group/RVfpga-sim-addons
RVfpga-Pipeline	https://github.com/artecs-group/RVfpga-sim-addons

Verilator is an open-source HDL (hardware description language) simulator that is used for compiling (or more precisely, “verilating” [16]) the RVfpga SoC, which then acts as the backend for several of the other simulation tools: RVfpga-Trace, RVfpga-ViDBo, and RVfpga-Pipeline. Because the backend simulates the RVfpga SoC’s actual source code, it is cycle-accurate. The frontend of each simulation tool is different. RVfpga-Trace makes use of the open-source GTKWave application to view signal traces throughout the SoC, allowing the user to dig into lower levels of the HDL hierarchy, which is especially useful when extending and debugging the system. We developed RVfpga-ViDBo, which allows users to simulate I/O programs on a **Vi**rtual **D**evelopment **Bo**ard [17] instead of having to purchase a costly FPGA board. The last tool, RVfpga-Pipeline, is an open-source tool for visualizing RVfpga’s 9-stage pipeline, which is especially useful when analyzing the internal VeeR EH1 microarchitecture in the last half of the labs.

Table 4 lists optional hardware: the Nexys A7-100T FPGA development board and its supporting tools. This board includes the Artix7 FPGA, peripherals, and interfaces targeted by the RVfpga system, including 7-segment displays, switches, LEDs, off-chip memory, and an accelerometer [18]. Users may synthesize the HDL and target this board using the Vivado WebPACK software, which is free. In 2023, the hardware cost $262 (academic price) to $349, but the hardware is optional; so, users can use and complete the entire RVfpga course in simulation and without cost, without the need to purchase the board. The RVfpga-ViDBo simulator targets and visualizes the Nexys A7 board in simulation, so many of the benefits of the Nexys A7 board are also already realized when using RVfpga-ViDBo, which is free.

Table 4. Optional Hardware and Tools

Tool	Website
Nexys A7-100T FPGA Board	https://store.digilentinc.com/nexys-a7-fpga-trainer-board-recommended-for-ece-curriculum/
Vivado 2019.2 WebPACK	https://www.xilinx.com/support/download/index.html/content/xilinx/en/downloadNav/vivado-design-tools/archive.html

Figure 4 shows the methods for running the RVfpga SoC: in hardware (RVfpga-Nexys) – using Vivado or downloading the configuration file (bitfile) in PlatformIO; or in software – using the simulation tools already discussed, RVfpga-VIDBo, RVfpga-Trace, and RVfpga-Pipeline, which use the Verilator simulator as a backend.

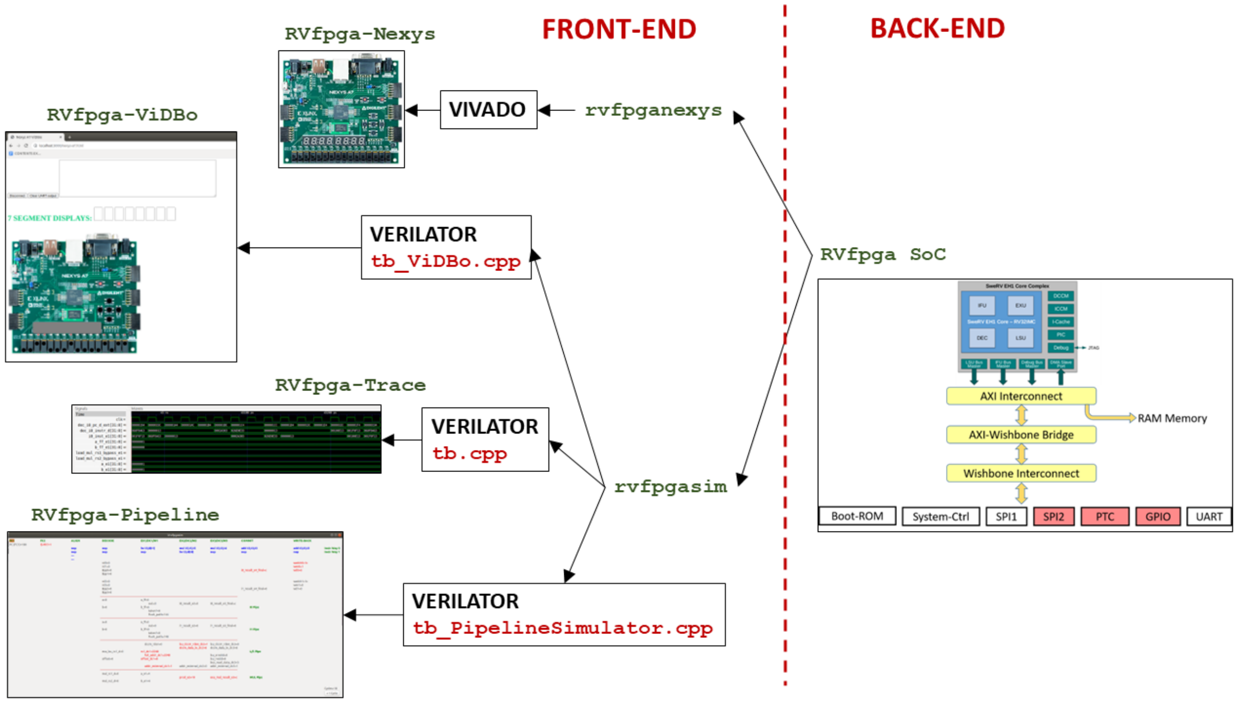

Figure 4. Map of RVfpga Tools

2.3 RVfpga Labs

The RVfpga course includes a Getting Started Guide, twenty labs, and supporting resources such as the Verilog/SystemVerilog source code for the entire RVfpga system and pre-generated bitfiles and simulation executables (for each of the 3 operating systems: Linux, Windows and MacOS) that contain the RVfpga SoC targeted for simulation or for use on the Nexys A7 board. Table 5 lists the twenty RVfpga labs. The first ten labs (Labs 1-10) comprise the first subcourse that focuses on programming and peripherals, and the second half (Labs 11-20) focus on the core's microarchitecture.

Labs 1-4 show how to program RVfpga using C, RISC-V assembly, and a combination of C and assembly. Lab 5, which is optional, shows how to create a Vivado project to generate a bitfile targeted to the Artix7 FPGA and the peripherals on the Nexys A7-100T board. This is necessary when users wish to extend the RVfpga SoC and test it in hardware, which requires a newly generated bitfile. Again, Lab 5 is optional because all labs, including system extension, can be completed in simulation only. Labs 6-10 introduce memory-mapped I/O and discuss existing peripherals within the SoC as well as how to extend the system to add more peripherals. Both programming and interrupt-based approaches are used in these labs. Interrupts are used with support from Western Digital's PSP (platform support package) and BSP (board support package), available at https://github.com/chipsalliance/riscv-fw-infrastructure; these are automatically installed within PlatformIO.

Labs 11-20 delve into the microarchitecture, including configuration settings, performance counters, the pipeline core, hazards, superscalar execution, and the memory hierarchy, including the ICCM and DCCM (instruction and data closely-coupled memories), and instruction cache – the underlying VeeR EH1 core does not include a data cache. Labs 11-20 also show how to modify the core and memory system, including exploring various core configurations and branch predictors and extending the core's capabilities, such as adding new instructions (specifically, we include instructions from several extensions not supported in the baseline system, such as bit manipulation and floating-point instructions) and additional performance counters.

Table 5. RVfpga Labs

Lab #	Title
Part 1	
0	RVfpga Labs Overview
1	C Programming
2	RISC-V Assembly Language
3	Function Calls
4	Image Processing: Projects with C & Assembly
5	Creating a Vivado Project (optional)
6	Introduction to I/O
7	7-Segment Displays
8	Timers
9	Interrupt-Driven I/O
10	Serial Buses
Part 2	
11	VeeR EH1 Configuration and Organization. Performance Monitoring
12	Arithmetic/Logical Instructions: the add instruction
13	Memory Instructions: the `lw` and `sw` instructions
14	Structural Hazards
15	Data Hazards
16	Control Hazards. Branch Instructions: the `beq` Instruction. The Branch.
17	Superscalar Execution
18	Adding New Features (Instructions, Hardware Counters) to the Core
19	Memory Hierarchy. The Instruction Cache.
20	ICCM and DCCM

In addition to the RVfpga course described in this paper, Imagination Technologies also provides a follow-on course called RVfpga-SoC that includes five labs the demonstrate how to (1) create the RVfpga SoC using Xilinx's Vivado to piece together building blocks including the VeeR EH1 core, interconnect, and peripherals; (2) run programs on this newly created RVfpga SoC; (3) use FuseSoC, an open-source tool for building systems, as an alternate method for creating the RVfpga SoC from building blocks; (4) build, run, and use Zephyr, an open-source real-time operating system (RTOS), on the RVfpga SoC system; and (5) build and run a Tensorflow Lite project on the system.

3. RVfpga MOOC

The RVfpga MOOC that we are developing in EdX includes 10 chapters that cover Labs 1-4, 6-9, and 11. The chapters include instructions, hands-on tutorials, videos (including demonstrations), exercises, and multiple-choice questions. The course begins with an overview of the course, the RVfpga system, and the tools (Chapter 1), including detailed instructions with videos that show how to install and use each of the tools listed in Table 3. After the introduction, the course covers nine RVfpga labs (1-4, 6-9 and 11), called chapters 2-10 in the EdX course. Each chapter includes at least two videos, one describing the principles and theoretical foundation of the topic covered in the chapter and one showing a demonstration of how to use and practice these principles. Each chapter also includes multiple choice questions and exercises for practicing and gain hands-on experience with the topics taught in the chapter. Like the full RVfpga course, the EdX course provides the entire source code (Verilog and SystemVerilog) for the RVfpga SoC. Course participants may use, modify, and simulate this source code throughout the course. Optionally, participants may also use the hardware, the Nexys A7 FPGA board, to test their modified RVfpga system and code. Because the course can be completed in simulation only (using the simulation tools listed in Table 3), the course may be completed without cost.

4. RVfpga Workshop

To increase accessibility to this RVfpga course, we have run ten international workshops since May 2022, four in the U.S., five in Europe, one in Israel, and two in Japan. These are "Train-the-Teacher" events and have been attended by 258 people. With guidance from the authors, these hands-on workshops enabled users to quickly run and use the RISC-V system and its tools. These 1-day workshops distill the most important aspects of the RVfpga package, describing the RVfpga system, showing attendees how to use and write code for it, and using hands-on practice to explore the peripherals and other features highlighted in the labs, including interrupts, performance counters, and benchmarking. Within the first hour of the workshop, users are able to write, compile, and run RISC-V programs on a commercial RISC-V core and SoC running in both hardware and simulation. Responses from attendees include:

"Excellent demonstration how real hardware and emulation matches, nice integration and IPC [instructions per cycle] and performance counter experiments."

"I enjoyed the practical knowledge about RVfpga. My first successful FPGA project!"

"A clear, focused, comprehensive [workshop] linking hardware, software and RISC-V and relevant computer architecture within a workable toolflow and affordable hardware options."

In 2023, we are continuing to run workshops and promote RVfpga adoption worldwide with workshops in Taiwan in July, multiple locations across China starting in September, and Scandinavia in Q4. In addition to the workshops, the authors hosted a Webinar sponsored by DigiKey to introduce RVfpga where 750 attended live, and more than 1,000 additional viewers subsequently watched the presentation [19].

5. Future Improvements

We plan on expanding the RVfpga course by:

- Targeting additional FPGA boards, especially lower-cost boards including those from vendors in addition to Xilinx,

- Using a smaller core, for example, the VeeR EL2 core. This core will be more appropriate for undergraduate students due to its simpler microarchitecture (it is a single-issue, 4-stage pipelined core). The smaller core and SoC that we will build around it will also fit on smaller, less expensive FPGAs.

These new features increase course accessibility, especially for those hoping to use RVfpga in hardware, by reducing the cost of the target FPGA board. It also allows users to access a smaller SoC that could be targeted to lower-cost embedded system applications and research.

6. Conclusions

We have developed the RVfpga course to increase understanding, accessibility, and usability of the RISC-V computer architecture, RISC-V SoCs and cores, and the RISC-V ecosystem, including the toolchain and simulators. This course is most typically implemented as a two-semester course but may also be used in a condensed 1-semester course or for self-study by industry professionals, academics, researchers, and others. The first half of the labs (1-10) is targeted toward a junior- or senior-level undergraduate course and the second half (Labs 11-20) is targeted to an upper-division or master's level course. The RVfpga course has already been licensed and downloaded by more than 2,800 users since its release in December 2020 and will likely exceed 4,000 by the end of 2023. We also expect the international RVfpga Workshops to exceed 500 attendees by the end of 2023. In addition, we are nearing completion of an RVfpga EdX MOOC that will increase accessibility to an even larger audience. In the next year, we plan on expanding the RVfpga course to include a smaller commercial RISC-V core and SoC that can be targeted to smaller, less expensive FPGA boards to enable users to implement the designs in hardware, where higher-cost FPGA boards may be prohibitive, and to facilitate understanding of the core microarchitecture.

Acknowledgments

We would like to acknowledge the support and contributions of David Patterson, Ted Marena, Zvonimir Bandic, Ivan Kravets, Valerii Koval, Roy Kravitz, Daniel León, Katzalin Olcoz, Alberto del Barrio, Fernando Castro, Manuel Prieto, Ataur Patwary, Christian Tenllado, Francisco Tirado, Román Hermida, Cathal McCabe, Dan Hugo, Braden Harwood, David Burnett, Gage Elerding, Brian Cruickshank, Deepen Parmar, Thong Doan, Oliver Rew, Niko Nikolay, and Guanyang He. We also acknowledge the support of projects PID2021-123041OB-I00, funded by MCIN/AEI/ 10.13039/501100011033 and by "ERDF A way of making Europe", and by the CM under grant S2018/TCS-4423.

References

[1] RISC-V International: https://riscv.org/. Accessed February 21, 2023.

[2] VeeR (SweRV) Cores: https://github.com/chipsalliance/Cores-VeeR-EH1, https://github.com/chipsalliance/Cores-VeeR-EL2, https://github.com/chipsalliance/Cores-VeeR-EH2. Accessed February 21, 2023.

[3] Arm Introduction to Computer Architecture: https://www.arm.com/resources/education/education-kits/computer-architecture. Accessed February 21, 2023.

[4] S. Harris, D. Harris, D. Chaver, R. Owen, Z. Kakakhel, E. Sedano, Y. Panchul, and B. Ableidinger, “MIPSfpga: Using a Commercial MIPS Soft-Core in Computer Architecture Education”. *IET Circuits, Devices & Systems*, 2017. 11.10.1049/iet-cds.2016.0383.

[5] RISC-V International University Resources: https://riscv.org/learn/. Accessed February 21, 2023.

[6] R. Agrawal, S. Bandara, A. Ehret, M. Isakov, M. Mark, and M. Kinsy, “The BRISC-V Platform: A Practical Teaching Approach for Computer Architecture”, *Proceedings of the Workshop on Computer Architecture Education*, pp. 1-8, Jun. 2019. 10.1145/3338698.3338891.

[7] N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti, R. Sen, K. Sewell, M. Shoaib, N. Vaish, M. D. Hill, and D. A. Wood. “The gem5 simulator”. *SIGARCH Computer Architecture News*, 39, 2, pp. 1-7. May 2011. DOI:https://doi.org/10.1145/2024716.2024718.

[8] Grenoble Institute of Technology. “LeaRnV: RISC-V based SoC Platform for Research Development and Education.” 2020. https://tima-amfors.gricad-pages.univ-grenoble-alpes.fr/learnv/. Accessed February 21, 2023.

[9] S. L. Harris, D. Chaver, L, Piñuel, J. I. Gomez-Perez, M. H. Liaqat, Z. L. Kakakhel, O. Kindgren, R. Owen. "RVfpga: Using a RISC-V Core Targeted to an FPGA in Computer Architecture Education," *31st International Conference on Field-Programmable Logic and Applications (FPL)*, Dresden, Germany, 2021, pp. 145-150, doi: 10.1109/FPL53798.2021.00032.

[10] Imagination Technologies – Teaching Resources: https://university.imgtec.com/teaching-download/. Accessed February 21, 2023.

[11] Imagination Technologies – RVfpga download: https://university.imgtec.com/rvfpga/. Accessed February 21, 2023.

[12] S. Harris and D. Harris. *Digital Design and Computer Architecture*. 2021. Morgan Kaufmann.

[13] VeeRwolf (SweRVolf) SoC - Master Branch: https://github.com/chipsalliance/Cores-SweRVolf. Accessed February 21, 2023.

[14] Programmer’s Reference Manual of VeeR EH1: https://github.com/chipsalliance/Cores-VeeR-EH1/tree/main/docs/RISC-V_VeeR_EH1_PRM.pdf. Accessed February 21, 2023.

[15] P.W.C. Prasad, A. Alsadoon, A. Beg, and A. Chan. “Using simulators for teaching computer organization and architecture”. Computer Applied Engineering Education, 24: 215-224. 2016. https://doi.org/10.1002/cae.21699.

[16] Verilator: https://www.veripool.org/wiki/verilator. Accessed February 21, 2023.

[17] ViDBo, https://github.com/olofk/vidbo. Accessed February 21, 2023.

[18] Nexys A7 Reference Manual: https://reference.digilentinc.com/_media/reference/programmable-logic/nexys-a7/nexys-a7_rm.pdf. Accessed February 21, 2023.

[19] S. Harris, “Understanding RISC-V Architecture and Implementation on an FPGA”. DigiKey-Sponsored Webinar. February 23, 2022. https://www.youtube.com/watch?v=ePv3xD3ZmnY. Accessed February 21, 2023.