RVfpga: Using a RISC-V Core Targeted to an FPGA in Computer Architecture Education

Sarah L. Harris
Dept. of Electrical & Computer Engineering
University of Nevada, Las Vegas
Las Vegas, USA
sarah.harris@unlv.edu

Daniel Chaver
Dept. of Computer Architecture and System Engineering
University Complutense of Madrid
Madrid, Spain
dani02@ucm.es

Luis Piñuel
Dept. of Computer Architecture and System Engineering
University Complutense of Madrid
Madrid, Spain
lpinuel@ucm.es

J.I. Gomez-Perez
Dept. of Computer Architecture and System Engineering
University Complutense of Madrid
Madrid, Spain
jigomez@ucm.es

M. Hamza Liaqat
AZKY Tech Labs
Islamabad, Pakistan
hamza@azkytech.com

Zubair L. Kakakhel
AZKY Tech Labs
Birmingham, United Kingdom
zubair@azkytech.com

Olof Kindgren
Qamcom Research & Technology
Gothenburg, Sweden
olof.kindgren@qamcom.se

Robert Owen
Imagination Technologies
Kings Langley, United Kingdom
robert.owen@imgtec.com

***Abstract*—RISC-V FPGA, also written RVfpga, is a set of two freely available courses developed by the authors and Imagination Technologies that enable users to understand and use the RISC-V instruction set architecture (ISA), a commercial RISC-V core and system, and the RISC-V ecosystem. The first course, RVfpga, includes comprehensive instructions, tools, and labs for targeting a commercial RISC-V processor to a field programmable gate array (FPGA) and then using and expanding it to learn about computer architecture, digital design, embedded systems, system-on-chip (SoC) design, and programming. The topics covered include targeting the RISC-V SoC to an FPGA, programming in C and RISC-V assembly, running programs in simulation or, optionally, in hardware, using peripherals and adding new ones to the SoC, and analyzing and modifying the RISC-V core and memory system, including adding new instructions to the core. The follow-on course, RVfpga-SoC, shows how to build a RISC-V SoC from building blocks and then run the Zephyr real-time operating system (RTOS) on it. At the completion of these courses, users will have a working RISC-V system and have hands-on experience exploring and using both the RISC-V SoC and the RISC-V toolchain, including compilers and simulators.**

Keywords—RVfpga, RVfpga-SoC, RISC-V, FPGA, SweRV, SweRVolf, computer architecture, education, system-on-chip, SoC

I. Introduction

RISC-V is an open-source instruction set architecture (ISA), introduced in 2010, that is becoming increasingly used in industry, academia, and research. While other open-source ISAs have existed, including OpenSPARC, which was developed in 2005, RISC-V is the first open-source architecture that has been widely adopted, thanks in part to RISC-V International [1], a group of industry and academic collaborators that ratify the RISC-V specifications and build the RISC-V community.

Because RISC-V is open source, the time-consuming and costly barrier of licensing is removed and collaboration is encouraged. However, the barrier of entry into using the tools and understanding the RISC-V core remains. This RISC-V FPGA course (RVfpga) removes these barriers by providing step-by-step instructions on topics ranging from installing and using the tools and programming in C and RISC-V assembly to expanding the processor to add peripherals such as serial interfaces and to modify the RISC-V core and memory system, including adding instructions to the core.

The *RVfpga Getting Started Guide* shows how to install and use the RISC-V toolchain to compile and run programs, how to program Digilent's Nexys A7 FPGA board with the RVfpga System, and how to run RISC-V programs in simulation using either Verilator or Western Digital's Whisper instruction set simulator (ISS). The twenty *RVfpga Labs* show how to program the core in C and RISC-V assembly, how to use peripherals and add new ones to the RVfpga System, and how to understand and modify the microarchitecture and memory system of the commercial, open-source SweRV EH1 core [2] provided by Western Digital. Finally, the *RVfpga-SoC* follow on course shows how to design, synthesize, and load a RISC-V SoC and how to run the open-source Zephyr real-time operating system (RTOS) on it.

This RVfpga course supports the open-source philosophy itself because the course materials are freely available upon request from Imagination Technologies [3]. In addition, all of the software tools are free, as well as the core and SoC, which are based on SweRV EH1 [2] and SweRVolf [4] respectively, both of which are open-source with an Apache 2.0 license. All of the materials can be completed in simulation, so the course may be completed without cost. The optional Nexys A7 FPGA board from Digilent Inc. that is used in the course costs $200, but many academics and researchers already have access to an FPGA board to which they could adapt the materials. (Note that the older Nexys 4 DDR board from Digilent can be used interchangeably with the Nexys A7 board.) Using the FPGA as the hardware target is optional, but it enhances the hands-on learning. Future versions of the RVfpga course will target lower-cost boards.

The rest of this paper provides an overview of the RVfpga course and materials (Section II), a description of the Getting Started Guide and Labs (Sections III and IV), and an overview of the RVfpga-SoC course (Section V). The paper concludes by describing related and future work (Sections VI and VII).

II. RVfpga Overview

RVfpga is a RISC-V computer architecture course developed by the authors and Imagination Technologies in collaboration with industry partners including Western Digital Corporation. The course enables users to gain hands-on experience with the RISC-V architecture and its ecosystem,

which includes the RISC-V toolchain, simulators, open-source hardware cores and SoCs, and software tools.

The course includes a comprehensive Getting Started Guide, slides, and twenty extensive labs. A stand-alone RVfpga-SoC course is also provided. The *RVfpga Getting Started Guide* and Labs 1-10 have been available since November 2020. The remaining labs, Labs 11-20, will be released in fall 2021, and the stand-alone RVfpga-SoC course is available as of summer 2021.

The target audience is anyone who wants to learn or teach about the RISC-V architecture and how to use it – including professors, industry professionals, researchers, and students. Before completing the RVfpga or RVfpga-SoC course, it is expected that users have at least some understanding of the following topics: digital logic design, high-level programming (preferably C), assembly programming, hardware description languages (HDL), instruction set architecture (ISA), input/output (I/O) systems, processor microarchitecture, and memory systems. These topics are covered in the textbook *Digital Design and Computer Architecture: RISC-V Edition*, Harris & Harris, © Morgan Kaufmann, 2021. Other textbooks, including *Computer Organization and Design RISC-V Edition*, Patterson & Hennessy, © Morgan Kaufmann 2017, cover some of this material. The RVfpga course develops and expands on these topics.

In the remainder of this section, we describe three cores and systems that are fundamental to the RVfpga System: the open-source commercial SweRV EH1 RISC-V core provided by Western Digital (Section IIa), the SweRVolfX SoC, an SoC that uses the EH1 core (Section IIb), and RVfpgaNexys/RVfpgaSim, which expand the SweRVolfX SoC (Section IIc). RVfpgaNexys targets the Nexys A7 FPGA board and RVfpgaSim is a wrapper that targets simulation.

A. *SweRV EH1 Core* and *SweRV EH1 Core Complex*

Western Digital has developed three open-source RISC-V cores over the past few years: SweRV EH1, SweRV EH2, and SweRV EL2. The **SweRV EH1 Core** (provided with the RVfpga package and also available from [2]) is a 32-bit, 2-way superscalar, 9-stage pipeline core that supports the multiply/divide (M) and compressed (C) RISC-V extensions. EH1 is preferred over EL2 or EH2 for its high performance/MHz and its simple thread structure, but future versions of RVfpga might also target the other cores.

We overview the SweRV EH1 core, but the Programmer's Reference Manual of the SweRV EH1 core [5] describes the core in detail. SweRV EH1 (see Figure 1) supports four arithmetic logic units (ALUs) in two pipelines: instruction way 0 (I0) and way 1 (I1). These ways support ALU operations, loads/stores and multiplications. The processor also has one out-of-pipeline 34-cycle latency divider. Four stall points exist in the pipeline in the following stages: Fetch 1, Align, Decode, and Commit. The Fetch 1 stage includes a Gshare branch predictor [6]. In the Align stage, instructions are retrieved from three fetch buffers. In the Decode stage, up to two instructions from four instruction buffers are decoded. In the Commit stage, up to two instructions per cycle are committed. Finally, in the Writeback stage, the architectural registers are updated.

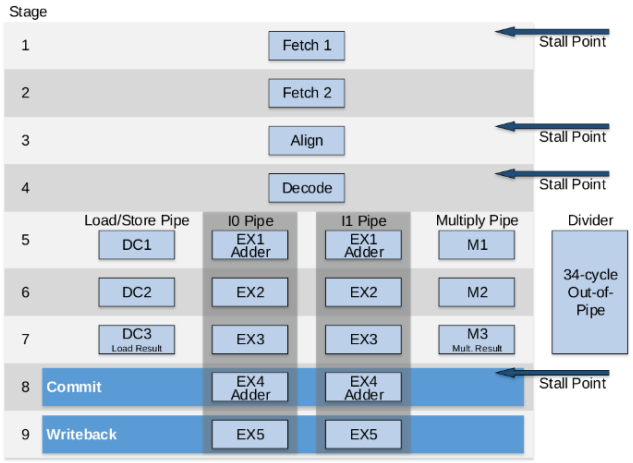

Fig. 1. SweRV EH1 core microarchitecture (figure from [5])

Figure 2 shows a comparison of existing commercial cores and processors. The SweRV EH1 Core performance per MHz is impressively high at 4.9 CM/MHz (CoreMark per MHz): it is twice as fast as the ARM Cortex A8 and its performance even surpasses the ARM Cortex A15 performance.

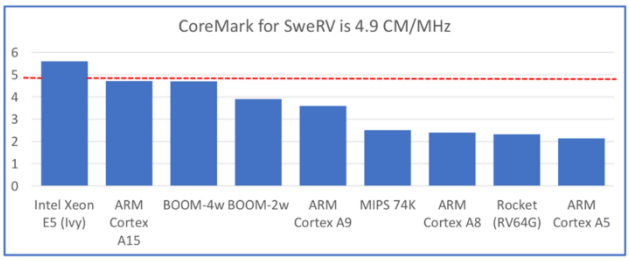

Fig. 2. Benchmark comparison per thread and MHz (figure from [7])

Western Digital also provides an extension to the SweRV EH1 Core called the **SweRV EH1 Core Complex** (see Figure 3), which adds the following elements to the EH1 Core:

- Two dedicated memories that are tightly coupled to the core: one for instructions (ICCM: instruction closely coupled memory), and the other for data (DCCM)..
- An optional 4-way set-associative instruction cache with parity or ECC protection.
- An optional Programmable Interrupt Controller (PIC), that supports up to 255 external interrupts.
- Four system bus interfaces for instruction fetch, data accesses, debug accesses, and external direct memory accesses (DMA) to closely coupled memories.
- Core Debug Unit.

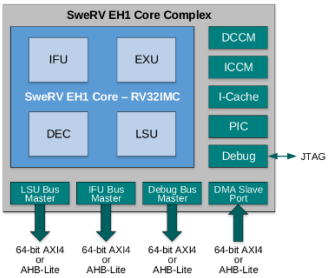

Fig. 3. SweRV EH1 Core Complex (figure from [5])

B. SweRVolf System on Chip and RVfpga Extensions

The SoC used in RVfpga is an extension of the SweRVolf SoC version 0.7.3, available at [8]. In RVfpga, we call this extended SoC **SweRVolfX** (SweRVolf eXtended). The SweRVolfX SoC is shown in Figure 4 as part of a bigger structure that we will describe in the next section.

The SweRVolfX SoC is built around the SweRV EH1 Core Complex shown in Figure 3. In addition, the SweRVolfX SoC also includes a Boot ROM, a UART, a System Controller, two SPI controllers, a GPIO and a Timer. Given that the SweRV EH1 Core uses an AXI bus and the peripherals use a Wishbone bus, the SweRVolfX SoC also has an AXI-to-Wishbone Bridge. Table I gives the memory-mapped addresses of the peripherals that are connected to the SweRV EH1 core via the Wishbone interconnect.

TABLE I. MEMORY-MAPPED ADDRESSES OF SWERVOLFX

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

* Peripherals added in SweRVolfX

C. RVfpgaNexys and RVfpgaSim

The SweRVolfX SoC can run either on the Nexys A7 (or Nexys4 DDR) FPGA board or in simulation. Figure 5 shows the hierarchical organization of the RVfpga System that we will describe further in this section.

RVfpgaNexys is the SweRVolfX SoC targeted to the Digilent Nexys A7 FPGA board (Figure 4). The main components of RVfpgaNexys are listed below and illustrated in Figure 4.

- RVfpga Hardware:
  - SweRVolfX SoC
  - Lite DRAM controller
  - Clock Generator: the Nexys A7 board includes a single 100 MHz crystal oscillator that is used by the Lite DRAM controller. The frequency of this clock is scaled down to 50 MHz to use in the SweRVolf SoC.
  - Clock Domain Crossing (CDC) module: connection of 2 clock domains: SweRVolf SoC and Lite DRAM.
  - BSCAN logic for the JTAG port
- Nexys A7 FPGA board memory and peripherals used by RVfpga:
  - DDR2 memory (accessed through the Lite DRAM controller)
  - USB connection
  - SPI Flash memory and SPI Accelerometer.
  - 16 LEDs and 16 Switches
  - 8-digit 7-Segment Displays

The Nexys A7 board [9] (or similarly the Nexys 4 DDR board) is a recommended trainer board for electrical and computer engineering curricula. This board costs $265 (or a discounted price of $199 with academic pricing). In future RVfpga versions we plan to include support for other Digilent boards such as Basys 3, Arty A7 or Cmod A7. Because these boards include FPGAs that are smaller than the Artix 7 FPGA that is on the Nexys A7 board, we will likely use the smaller SweRV EL2 core for these smaller boards.

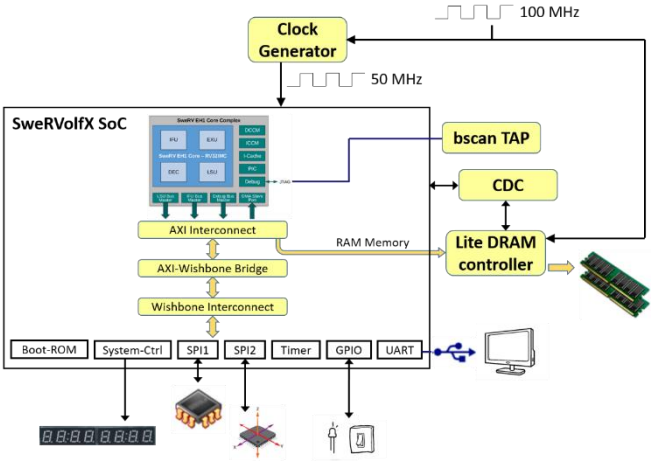

Fig. 4. RVfpgaNexys

RVfpgaSim is the SweRVolfX SoC wrapped in a testbench to be used by HDL simulators. In the RVfpga course, we show how to use Verilator [10]. This open-source and free HDL simulator claims to be the fastest Verilog/SystemVerilog simulator; it is widely used in industry and academia; it provides out-of-the-box support from ARM and RISC-V vendor IPs; and it is guided by Chips Alliance and the Linux Foundation.

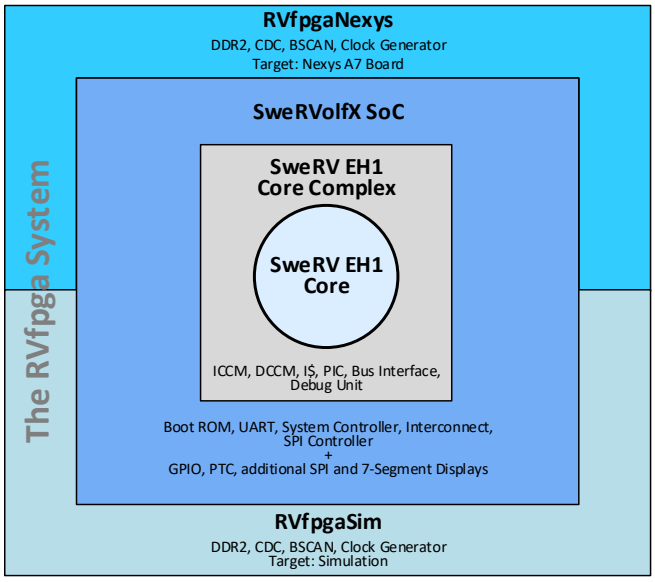

Fig. 5. The RVfpga System Hierarchy

D. Hardware and Software Requirements

The software required by the RVfpga course is all freely available and the hardware is optional, as listed in Table II. RVfpga requires two main software tools: PlatformIO, which is an extension of Visual Studio (VS) Code, and Verilator. The other software will either be installed as part of other installations or is optional. RVfpga can be used on Windows, Linux, or macOS platforms.

The RVfpga System source code, which includes the SweRVolfX and the SweRV EH1 Core Complex, as well as the RVfpgaNexys and RVfpgaSim wrappers, is provided with the RVfpga course materials, but links to the GitHub distributions of SweRVolf and the EH1 core are provided for

reference in Table II. Descriptions of how to install the needed software and use the RVfpga System are provided in the RVfpga Getting Started Guide, which is discussed next.

TABLE II. RVFPGA SOFTWARE AND HARDWARE

Software		
Name	**Website**	**Cost**
Vivado 2019.2 WebPACK*	https://www.xilinx.com/support/download/index.html/content/xilinx/en/downloadNav/vivado-design-tools/2019-2.html	free
VS Code	https://code.visualstudio.com/Download	free
PlatformIO	https://platformio.org/ Installed within VSCode	free
Verilator (an HDL simulator) and GTKWave	https://github.com/verilator/verilator http://gtkwave.sourceforge.net/	free
Whisper (Western Digital's RISC-V Instruction Set Simulator)	https://github.com/chipsalliance/SweRV-ISS Installed within PlatformIO	free
RISC-V Toolchain and OpenOCD	https://github.com/riscv/riscv-gnu-toolchain https://github.com/riscv/riscv-openocd Installed within PlatformIO	free
Hardware		
Name	**Website**	**Cost**
Nexys A7 FPGA Board*	https://store.digilentinc.com/nexys-a7-fpga-trainer-board-recommended-for-ece-curriculum/	$265 (academic price: $199)
RISC-V Core and System-on-Chip (SoC)		
Name	**Website**	**Cost**
Western Digital's SweRV EH1 Core	https://github.com/chipsalliance/Cores-SweRV	free
SweRVolf	https://github.com/chipsalliance/Cores-SweRVolf	free

* optional

III. RVFPGA GETTING STARTED GUIDE

The RVfpga Getting Started Guide (GSG) is the introductory document of the RVfpga materials. It is an extensive document that includes 8 sections plus appendices.

The Quick Start Guide, which is Section 2 of the GSG, describes the minimal software installation needed for RVfpga and then shows how to download and execute a simple example program on the RVfpga System, both in simulation and hardware.

Sections 3 and 4 give a brief introduction to the RISC-V computer architecture and the RVfpga System, including the organization of the Verilog and SystemVerilog files that make up the RVfpga System.

The remaining sections of the GSG show how to use the RVfpga System in both hardware (RVfpgaNexys) and simulation (RVfpgaSim). Section 5 shows how to install the software tools needed to use the RVfpga System. Section 6 shows how to use PlatformIO to both download RVfpgaNexys onto the Nexys A7 FPGA board and how to download and run several example programs on it. Sections 7 and 8 show how to simulate RVfpgaSim using Verilator and how to simulate RISC-V code on the Whisper instruction set simulator (ISS), respectively.

IV. RVFPGA LABS

The twenty RVfpga laboratory assignments provided with the RVfpga course are listed in TABLE III. In this section we describe each lab and highlight each lab's objectives.

TABLE III. RVFPGA LABS

	#	Title
Part 1	0	RVfpga Labs Overview
	1	Creating a Vivado Project
	2	C Programming
	3	RISC-V Assembly Language
	4	Function Calls
	5	Image Processing: Projects with C & Assembly
	6	Introduction to I/O
	7	7-Segment Displays
	8	Timers
	9	Interrupt-Driven I/O
	10	Serial Buses
Part 2	11	SweRV 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

Lab 0 overviews all of the labs. Each lab includes instructions that introduce and describe the topics and exercises for guiding the users in then experimenting with the concepts on their own. The RVfpga package also includes solutions for all exercises.

Lab 1 shows how to use Xilinx's Vivado to target the RVfpga System to the Artix 7 FPGA on the Nexys A7 FPGA board. When the user modifies the RVfpga System in labs 6-20, they will rely on what they learned from Lab 1 to resynthesize and debug their modified RVfpga systems.

Labs 2 through 5 show how to write C and RISC-V assembly programs and run them – either in simulation (Whisper) or in hardware using PlatformIO and the Nexys A7 board. Lab 2 shows how to write, compile, debug, simulate, and run C programs on the RVfpga System. PlatformIO offers a seamless environment for doing this. Lab 2 also shows how to access the peripherals (such as the LEDs and switches) in C programs and how to use PlatformIO's serial monitor to view a program's print output.

Lab 3 shows how to write and run RISC-V assembly programs. Lab 4 shows how to use function calls, including C library functions. Lab 5 shows how to create a project that uses both C and assembly source files and how to perform image processing algorithms.

Labs 6 through 10 show how to use the peripherals available in the RVfpga System (i.e., the switches, LEDs, seven-segment displays, timer, and SPI accelerometer) and also how to modify it to add new peripherals (pushbuttons and tri-color LEDs). Users are guided to modify the RVfpga

System source code (Verilog/SystemVerilog code) and also to write programs to exercise their added hardware. Lab 9 introduces the concept of interrupts and shows how to use them in the RVfpga System.

Labs 11 through 20, which will be released in fall 2021, show how to investigate, analyze, and modify the RISC-V core and memory system. Lab 11 provides a detailed introduction to the SweRV EH1 microarchitecture, explaining each of its 9 pipeline stages and each of the execution pipes. This lab also shows how to use the performance counters and real benchmarks to test the RVfpga System. Labs 12 and 13 investigate arithmetic/logic and memory (load and store) instructions. Labs 14-16 investigate structural/data/control hazards, branch instructions, and the (Gshare) branch predictor. Lab 17 analyzes 2-way superscalar execution. Lab 18 shows how to add new instructions to the core, including bit manipulations and floating-point instructions, as well as new hardware counters. Labs 19 and 20 show how to use the instruction cache and the closely-coupled instruction and data memories (ICCM and DCCM).

V. RVfpga-SoC Course

While the RVfpga course focuses on using and expanding the RISC-V core and its peripherals, the RVfpga-SoC course, available in summer 2021, shows how to create a RISC-V SoC, based on SweRVolfX, from building blocks.

The five RVfpgaSoC laboratory assignments are listed in TABLE IV. Lab 1 gives an overview and shows how to create the RVfpga SoC from scratch by connecting modules including the SweRV EH1 core, interconnect, and peripherals. We use Xilinx's Vivado Block design tool to both add modules and then connect them pin-by-pin to create a subset of the RVfpga System. We then show how to target this RVfpga System to the Nexys A7 FPGA board. Lab 2 shows how to run programs on the RVfpga System built in Lab 1.

TABLE IV. RVfpga-SoC Labs

#	Title
1	Introduction to RVfpga-SoC
2	Running Software on the RVfpga SoC
3	Introduction to SweRVolf and FuseSoC
4	Building and Running Zephyr on the SweRVolf
5	Running Tensorflow Lite on SweRVolf

Users may run the programs either in simulation or in hardware. Lab 3 introduces FuseSoC, an open-source tool for building systems, and SweRVolf, the open-source SoC based on the SweRV EH1 core. Lab 3 also compares the FuseSoC approach with the block design approach from Lab 1. Lab 4 shows how to build, run, and use Zephyr, an open-source real-time operating system (RTOS), on the RVfpga System. Finally, Lab 5 shows how to build a Tensorflow Lite project for Zephyr (a real-time operating system) and then run that Zephyr program on SweRVolf.

VI. Related Work

RVfpga fulfills the ACM's (Association from Computing Machinery's) Computing two main computer engineering competencies established in its Curricula 2020 for computer architecture and organization: "Manage the design of computer hardware components and integrate such components" and "Simulate and evaluate the performance of ... hardware solutions" [11].

Traditionally, undergraduate and graduate architecture courses relied extensively on simulators [12], such as gem5 [13] or Mars [14], which allow the students to evaluate the performance of different architectural choices. Architectural simulators are not too hard to set up and, to a certain extent, are highly configurable, which enables rapid exploration of an architecture. However, they are far from real systems.

Recently, HDL-based environments have arisen, offering a more detailed view of the core internals but with the trade-off of more complexity and decreased flexibility. However, these HDL systems offer a powerful alternative for computer architecture labs. Two such systems and accompanying programs are based on MIPS [15] and ARM [16] processors. MIPSfpga, introduced in 2015, is an architecture course built around a soft-core commercial MIPS processor [15] that includes labs on computer architecture covering topics similar to those in RVfpga. In 2020, the Arm University Program launched its "Arm-based Computer Architecture Education kit" [16] using a simplified non-commercial Arm core.

In addition to these existing programs, the emergence of the RISC-V architecture has led to a plethora of open-source educational materials [17] and design alternatives, ranging from very simple in-order designs to more aggressive out-of-order proposals. Two such designs are the BRISC-V Toolbox [18] and LeaRnV [19]. BRISC-V represents a huge effort to build a complete web-based ecosystem including a compiler, simulator, and a generator of fully-synthesizable hardware systems. However, this toolbox provides only the cores but not accompanying labs or exercises. The LeaRnV course aims to train students in hardware-software co-design.

VII. Conclusions and Future Work

We have developed two courses, RVfpga and RVfpga-SoC, that show how to target a commercial RISC-V core and system to an FPGA and to a simulator, use and modify the system, and use RISC-V tools such as compilers and debuggers. RVfpga, the first course, shows how to use, modify, and extend the system, including adding new peripherals and instructions, and analyzing the core microarchitecture and the cache and memory system by means of different mechanisms (Verilator simulations, performance counters and other tools). The second course, RVfpga-SoC, shows how to build a subset of the RVfpga System from scratch and how to run the Zephyr RTOS on it.

In the future, we plan on developing online and in-person workshops as well as a self-guided MOOC (massive open online course) based on this RVfpga course, both in English and Chinese. These will include videos and demonstrations to guide users through the materials. RVfpga written materials are currently available in English and Chinese (simplified and traditional) and will soon be translated into Spanish, Turkish, Japanese, and Korean. As mentioned, we also aim to target RVfpga to smaller FPGAs found on less expensive boards, such as the Basys 3, Arty A7, or Cmod A7 boards, by using one of Western Digital's smaller RISC-V open cores such as the SweRV EL2.

Acknowledgments

We would like to acknowledge the support and contributions of David Patterson, Ted Marena, 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 would also like to acknowledge the support of project MINECO RTI2018-093684-B-I00.

REFERENCES

[1] RISC-V International: https://riscv.org/
[2] SweRV EH1 Core: https://github.com/chipsalliance/Cores-SweRV
[3] Imagination Technologies – Teaching Resources: https://university.imgtec.com/teaching-download/
[4] SweRVolf SoC – Master Branch: https://github.com/chipsalliance/Cores-SweRVolf
[5] Programmer's Reference Manual of SweRV EH1: https://github.com/chipsalliance/Cores-SweRV/blob/master/docs/RISC-V_SweRV_EH1_PRM.pdf
[6] S.McFarling. "Combining branch predictors". Technical note TN-36, DEC-WRL, 1993.
[7] SweRV Cores Roadmap: https://content.riscv.org/wp-content/uploads/2019/12/12.11-14.20a3-Bandic-WD_SweRV_Cores_Roadmap_v4SCR.pdf
[8] SweRVolf SoC – Version 0.7.3: https://github.com/chipsalliance/Cores-SweRVolf/releases/tag/v0.7.3
[9] Nexys A7 Reference Manual: https://reference.digilentinc.com/_media/reference/programmable-logic/nexys-a7/nexys-a7_rm.pdf
[10] Verilator: https://www.veripool.org/wiki/verilator
[11] ACM/IEEE Computing Curricula 2020: https://www.acm.org/binaries/content/assets/education/curricula-recommendations/cc2020.pdf
[12] Prasad, P.W.C., Alsadoon, A., Beg, A. and Chan, A. (2016), "Using simulators for teaching computer organization and architecture". Comput Appl Eng Educ, 24: 215-224. https://doi.org/10.1002/cae.21699.
[13] Nathan Binkert, Bradford Beckmann, Gabriel Black, Steven K. Reinhardt, Ali Saidi, Arkaprava Basu, Joel Hestness, Derek R. Hower, Tushar Krishna, Somayeh Sardashti, Rathijit Sen, Korey Sewell, Muhammad Shoaib, Nilay Vaish, Mark D. Hill, and David A. Wood. 2011. The gem5 simulator. SIGARCH Comput. Archit. News 39, 2 (May 2011), 1–7. DOI:https://doi.org/10.1145/2024716.2024718.
[14] K. Vollmar and P. Sanderson. "Mars: An education-oriented MIPS assembly language simulator". In Proceedings of the 37th SIGCSE Technical Symposium on Computer Science Education, SIGCSE '06, pages 239–243, New York, NY, USA, 2006. ACM.
[15] Harris, Sarah & Harris, David & Chaver, Daniel & Owen, Robert & Kakakhel, Zubair & Sedano Algarabel, Enrique & Panchul, Yuri and Ableidinger, Bruce. (2017). "MIPSfpga: Using a Commercial MIPS Soft-Core in Computer Architecture Education". IET Circuits, Devices & Systems. 11. 10.1049/iet-cds.2016.0383.
[16] Arm Introduction to Computer Architecture: https://www.arm.com/resources/education/education-kits/computer-architecture
[17] RISC-V International University Resources: https://riscv.org/community/learn/educational-materials
[18] Agrawal, Rashmi & Bandara, Sahan & Ehret, Alan & Isakov, Mihailo & Mark, Miguel & Kinsy, Michel. (2019). The BRISC-V Platform: A Practical Teaching Approach for Computer Architecture. 1-8. 10.1145/3338698.3338891.
[19] Grenoble Institute of Technology. (2020). LeaRnV: RISC-V based SoC Platform for Research Development and Education. https://tima-amfors.gricad-pages.univ-grenoble-alpes.fr/learnv/