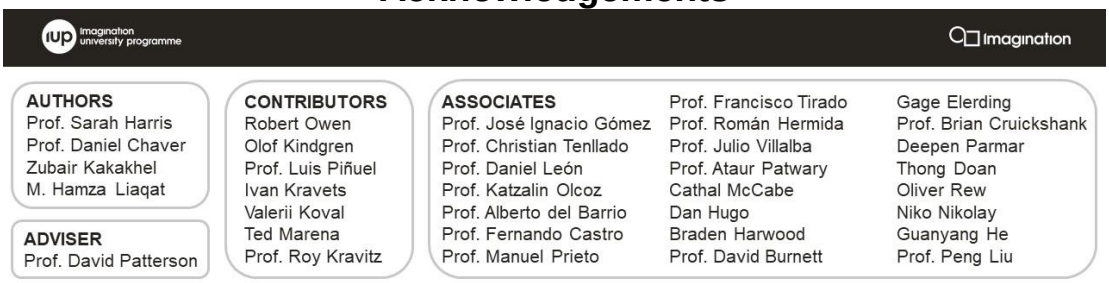
THE IMAGINATION UNIVERSITY PROGRAMME

RVfpga
Workshop Guide

Acknowledgements



Sponsors and Supporters

AUTHORS

- Prof. Sarah Harris (https://www.linkedin.com/in/sarah-harris-12720697/)
- Prof. Daniel Chaver (https://www.linkedin.com/in/daniel-chaver-a5056a156/)
- Zubair Kakakhel (https://www.linkedin.com/in/zubairlk/)
- M. Hamza Liaqat (https://www.linkedin.com/in/muhammad-hamza-liaqat-ab73a0195/)

ADVISER

- Prof. David Patterson (https://www.linkedin.com/in/dave-patterson-408225/)

CONTRIBUTORS

- Robert Owen (https://www.linkedin.com/in/robert-owen-4335931/)
- Olof Kindgren (https://www.linkedin.com/in/olofkindgren/)
- Prof. Luis Piñuel (https://www.linkedin.com/in/lpinuel/)
- Ivan Kravets (https://www.linkedin.com/in/ivankravets/)
- Valerii Koval (https://www.linkedin.com/in/valeros/)
- Ted Marena (https://www.linkedin.com/in/tedmarena/)
- Prof. Roy Kravitz (https://www.linkedin.com/in/roy-kravitz-4725963/)

ASSOCIATES

- Prof. José Ignacio Gómez (https://www.linkedin.com/in/jos%C3%A9-ignacio-gomez-182b981/)
- Prof. Christian Tenllado (https://www.linkedin.com/in/christian-tenllado-31578659/)
- Prof. Daniel León (www.linkedin.com/in/danileon-ufv)
- Prof. Katzalin Olcoz (https://www.linkedin.com/in/katzalin-olcoz-herrero-5724b0200/)
- Prof. Alberto del Barrio (https://www.linkedin.com/in/alberto-antonio-del-barrio-garc%C3%ADa-1a85586a/)
- Prof. Fernando Castro (https://www.linkedin.com/in/fernando-castro-5993103a/)
- Prof. Manuel Prieto (https://www.linkedin.com/in/manuel-prieto-matias-02470b8b/)
- Prof. Francisco Tirado (https://www.linkedin.com/in/francisco-tirado-fern%C3%A1ndez-40a45570/)
- Prof. Román Hermida (https://www.linkedin.com/in/roman-hermida-correa-a4175645/)
- Prof. Julio Villalba (https://www.linkedin.com/in/julio-villalba-moreno-97474824)
- Cathal McCabe (https://www.linkedin.com/in/cathalmccabe/)
- Dan Hugo (https://www.linkedin.com/in/danhugo/)
- Braden Harwood (https://www.linkedin.com/in/braden-harwood/)
- David Burnett (https://www.linkedin.com/in/david-burnett-3b03778/)
- Gage Elerding (https://www.linkedin.com/in/gage-elerding-052b16106/)
- Brian Cruickshank (https://www.linkedin.com/in/bcruiksh/)
- Deepen Parmar (https://www.linkedin.com/in/deepen-parmar/)
- Thong Doan (https://www.linkedin.com/in/thong-doan/)
- Oliver Rew (https://www.linkedin.com/in/oliver-rew/)
- Niko Nikolay (https://www.linkedin.com/in/roy-kravitz-4725963/)
- Guanyang He (https://www.linkedin.com/in/guanyang-he-5775ba109/)
- Prof. Ataur Patwary (https://www.linkedin.com/in/ataurpatwary/)
- Prof. Peng Liu (https://person.zju.edu.cn/liupeng)

"*RISC-V is transforming processor design and software/hardware co-design. RISC-V is an open architecture, which enables open-source hardware implementations. This new option means that software development can occur alongside hardware development, accelerating the design path. The RVfpga course enhances the understanding of not only RISC-V processors but also the RISC-V ecosystem and RISC-V SoCs. This course provides a deep understanding of an industrial-strength processor architecture and system of increasing popularity, which will prove useful throughout their academic and industry careers.*"

– David Patterson, University of California, Berkeley

1. OVERVIEW

This RVfpga Workshop Guide gives a guideline of a one-day workshop on RVfpga. Attendees may follow this guide in tandem with the workshop, and workshop instructors may use it to guide the workshop. This guide includes the following:

- **Outline** of the workshop materials that lists the topics and associated slide numbers (from RVfpga_Slides.pptx), demonstrations, and hands-on activities. An instructor may choose to include only a subset of the hands-on and demonstration (demo) activities.
- **Suggested Workshop Schedule** for a one-day workshop. Note that although the workshop is organized to last one day, some of the demonstrations and hands-on aspects may be removed to convert it to a half-day workshop or even a several-hour workshop, as needed.
- **Demos & Hands-On Details** about each outline topic that give additional information for completing the demonstrations and hands-on activities.

To take full advantage of the workshop, attendees should request the RVfpga Package from Imagination Technologies (https://university.imgtec.com/rvfpga/) prior to attending the RVfpga Workshop. This Workshop Guide is included as part of the RVfpga Package in the RVfpga\Documents folder. Workshop instructors may choose to provide this guide to attendees as printed material as well so that they can easily follow along during the workshop.

RVfpga Workshop Guide Contents:

2. WORKSHOP OUTLINE

Part 0: Introduction, Installation, Programming & Simulation

#	Topic	Slides	Demo	Hands-On
Introduction & Installation				
A	Introduction & Overview	1-31		
B	RVfpga Minimal Installation	33-34		GSG Section 2.A. Install VSCode, PlatformIO & GTKWave
C	Install Chips Alliance Platform & Run Example Program on Board	36		GSG Section 2.B. Use PlatformIO to run LEDsSwitches on FPGA board
Programming & Simulation				
D	AL_Operations: Run & Debug on Board	37		GSG Section 6.B: follow instructions to run/debug program on the board: AL_Operations
E	AL_Operations: Simulate in Whisper	38		GSG Section 8: follow instructions (skip Step 5 - will do later)
F	AL_Operations: Simulate in Verilator	39		GSG Section 7: follow instructions (skip simulation binary generation)
G	HelloWorld: Run on Board	40		GSG Section 6.F: follow instructions
H	HelloWorld: Simulate in Whisper	41		GSG Section 8: follow instructions (include Step 5)

Part 1: RVfpga Labs – Part 1 – Labs 1 to 10: Programming & I/O

#	Topic	Slides	Demo	Hands-On
RVfpga Labs – Part 1				
1	Lab 1: C Programming	42-49	Exercise 4 & 9	
2	Lab 2: Assembly Programming	50-56		Create PlatformIO Project (follow Section 3)
3	Lab 3: Function Calls	57-70	Exercise 6	
4	Lab 4: Image Processing – C & Assembly	71-70	Example: ImageTransformation	
5	Lab 5: Vivado Project	78-79	Create a Vivado Project – Section 2	
6	Lab 6: I/O Introduction	80-91		Exercise 5
7	Lab 7: 7-Segment Displays	92-101		Exercise 1
8	Lab 8: Timers	102-110	Exercises 1 & 3	
9	Lab 9: Interrupt-Driven I/O	111-119	Example – Section 5: 1. LED-Switch_C-Lang 2. 7SegDispl_C-Lang 3. 7SegDispl_Interrupts _C-Lang	Exercise 1
10	Lab 10: Serial Buses	120-130	Exercises 1 & 3	

Part 2: RVfpga Labs – Part 2 – Labs 11 to 20: Core & Memory System

#	Topic	Slides	Demo	Hands-On
RVfpga Labs – Part 2				
11	Lab 11: SweRV EH1 Configuration & Performance Monitoring	134-152		Section 2.D – Figure 11 Section 3.B – Figure 13
12	Lab 12: `add`	153-161	Exercise 7	
13	Lab 13: `lw` and `sw`	162-179		
14	Lab 14: Structural Hazards	180-190		Task (in Section 2.A: MUL_Instruction)
15	Lab 15: Data Hazards	191-208	Tasks (in Sections 2.A & 3)	
16	Lab16: Control Hazards and Branches	209-220		Task (in Section 3)
17	Lab 17: Superscalar Execution	221-235	Exercises 3, 5 and 6	
18	Lab 18: Adding New Features	236-237		Exercises 1, 3 and 4
19	Lab 19: I$	238-255		
20	Lab 20: ICCM, DCCM & Benchmarking	256-270		Section 3.A, 3.B and 3.C

3. SUGGESTED WORKSHOP SCHEDULE

Time	Topic #	Topic
09:00-09:30	A	RVfpga Introduction
09:30-10:00	B-C	RVfpga Tools Installation
10:00-10:30	D-H	Programming & Simulation
10:30-11:00	1-4	RVfpga Labs 1-4: C & Assembly Programming
11:00-11:15		*Break*
11:15-11:30	5	RVfpga Lab 5: RVfpga Vivado Project
11:30-12:30	6-10	RVfpga Labs 6-10: RVfpga I/O
12:30-13:15		*Lunch*
13:15-14:45	11-16	RVfpga Labs 11-16: Core Configuration & Pipeline
14:45-15:00		*Break*
15:00-16:00	17-20	RVfpga Labs 17-20: Superscalar, Memory & Benchmarking
16:00-17:00		Wrap-up & Questions

4. DEMOS & HANDS-ON DETAILS

PART 0. Introduction, Installation, Programming & Tools

A. Introduction

No hands-on or demo.

B. RVfgpa Minimal Installation

HANDS ON:

1. Install VSCode and PlatformIO:

Follow the instructions from Section 2.A of the Getting Started Guide ("*Minimal installation: VSCode, PlatformIO and Nexys A7 board drivers*").

   a. Install VSCode
   b. Install PlatformIO on top of VSCode
   c. Install Nexys A7 cable drivers:
      i. In Linux follow the instructions at the end of this section.
      ii. In Windows follow the instructions from Appendix B of the GSG.
      iii. In MacOS this is not necessary.

2. Install GTKWave:

   a. **Linux**: Open a terminal, and install GTKWave:
      - `sudo apt-get install -y gtkwave`

   b. **Windows**: GTKWave can be downloaded as a precompiled package from https://sourceforge.net/projects/gtkwave/files/. Look for the most recent Windows package, and download, execute and use it in your Windows machine.

   c. **MacOS**: You will use Homebrew to install *gtkwave*. But this time we need to use *cask* because it is a GUI macOS application. Type the following commands in an open Terminal:

      - `brew tap homebrew/cask`
      - `brew cask install xquartz`
      - `brew cask install gtkwave`

      After the installation, an icon for *gtkwave.app* should appear in the Application folder. In order to use it from the command line, you may need to install Perl's Switch module:

      - `cpan install Switch`

C. Install ChipsAlliance Platform and Run Example Program (LEDsSwitches) on the Nexys-A7-100T FPGA board

HANDS ON: Follow the instructions from Section 2.B of the Getting Started Guide ("*Download RVfpgaNexys onto FPGA and run programs on RVfpgaNexys*"), as summarized below:

Step 1. Connect Nexys A7 FPGA board to computer and turn the board on
Step 2. Open PlatformIO and C program
Step 3. Download RVfpgaNexys to Nexys A7 board
Step 4. Download and run program LEDsSwitches on RVfpgaNexys

D. Run & Debug Program (AL_Operations) on the FPGA board

HANDS ON: Follow the instructions from Section 6.B of the Getting Started Guide ("*AL_Operations program*").

E. Simulate Program AL_Operations in Whisper

HANDS ON: Follow the instructions from Section 8 of the Getting Started Guide ("*Simulation in Whisper*"). Skip step 5, which refers to the HelloWorld example that you will test later.

F: Simulate Program AL_Operations in Verilator

HANDS ON: Follow the instructions from Section 7 of the Getting Started Guide ("*Simulation in Verilator*").

Step 1. GENERATE THE SIMULATION BINARY, Vrvfpgasim:

- THIS STEP IS NOT NECESSSARY IN THE WORKSHOP!
- Binaries are provided at:
  [RVfpgaPath]/RVfpga/verilatorSIM/OriginalBinaries

Step 2. GENERATE THE SIMULATION TRACE FROM PLATFORMIO, USING Vrvfpgasim

Step 3. ANALYSE THE SIMULATION TRACE IN GTKWAVE: For showing the signals in GTKWave, do not follow steps 10 and 11, but use file *test.tcl* as explained in step 12.

G. Run Program HelloWorld on the Board

HANDS ON: Follow the instructions from Section 6.F of the Getting Started Guide ("*HelloWorld_C-Lang program*").

Note that in Linux you will have to configure the system as explained at the beginning of the aforementioned section of the GSG.

H. Simulate Program HelloWorld in Whisper

HANDS ON: Follow the instructions from Section 8. In this case you have to replace in the instructions “AL_Operations” for “HelloWorld” and take into consideration the explanations of Step 5.

PART 1. RVfpga Labs – Part 1 – Labs 1 to 10

- Instructions and programs for the RVfpga Labs are available in the following folder in a subfolder with the lab name (i.e., “Lab0”, “Lab1”, etc.):
  [RVfpgaPath]/RVfpga/Labs/

- Solutions used in the RVfpga Labs are available at:
  [RVfpgaPath]/RVfpga/Labs/RVfpgaLabsSolutions/ProgramsAndDocuments

- Bitstreams available at:
  Original:
  [RVfpgaPath]/RVfpga/src/rvfpganexys.bit
  Modified in Labs 6-10 and Lab 18:
  [RVfpgaPath]/RVfpga/Labs/RVfpgaLabsSolutions/Modified_RVfpgaSystem

- Verilator binaries available at:
  [RVfpgaPath]/RVfpga/verilatorSIM/OriginalBinaries

1. Lab 1 – C Programming

DEMO: Exercise 4. PlatformIO project available at:
*[RVfpgaPath]/RVfpga/Labs/RVfpgaLabsSolutions/ProgramsAndDocuments/Lab01/**4bitAdd***

DEMO: Exercise 9. PlatformIO project available at:
*[RVfpgaPath]/RVfpga/Labs/RVfpgaLabsSolutions/ProgramsAndDocuments/Lab01/**BubbleSort***

2. Lab 2 – Assembly Programming

HANDS ON: Create a PlatformIO project from scratch. Instructions in Section 3 of the document for Lab 2.

3. Lab 3 – Function Calls

DEMO: Exercise 6.

- This program uses functions to modularize the program and standard C libraries (such as *srand()* to generate random numbers).
- In other exercises, we also use the functions from WD’s PSP/BSP (such as *printfNexys()*).
- PlatformIO project available at:
  *[RVfpgaPath]/RVfpga/Labs/RVfpgaLabsSolutions/ProgramsAndDocuments/Lab03/**SimonSays***

4. Lab 4 – Image Processing – C and Assembly

DEMO: Image Transformation: Show original colour image and transformed grey image. PlatformIO project available at:

*[RVfpgaPath]/RVfpga/Labs/Lab04/**ImageTransformation***

5. Lab 5 – Vivado Project

DEMO: Section 2: Create a Vivado project and generate bitstream for RVfpgaNexys. Leave Vivado working on the background and, once done, test the bitstream with the LedsSwitches program.

6. Lab 6 – Introduction to I/O – GPIO

DEMO: Review the LedsSwitches example from the GSG and Exercise 4 from Lab 1.

HANDS ON: Exercise 5:

- For executing this program you must use the modified bitstream with support for the five on-board pushbuttons (as guided in Exercises 3 and 4), which is provided at:
  [RVfpgaPath]/RVfpga/Labs/RVfpgaLabsSolutions/Modified_RVfpgaSystem/RVfpgaSystem_Labs6-10/src/rvfpganexys.bit
- Use the PlatformIO project available at:
  *[RVfpgaPath]/RVfpga/Labs/RVfpgaLabsSolutions/ProgramsAndDocuments/Lab06/**Exercise5***

7. Lab 7 – 7-Segment Displays

HANDS ON: Exercise 1:

- For executing this program you must use the original bitstream again, which is provided at:
  [RVfpgaPath]/RVfpga/src/rvfpganexys.bit
- Use the PlatformIO project available at:
  *[RVfpgaPath]/RVfpga/Labs/RVfpgaLabsSolutions/ProgramsAndDocuments/Lab07/**Exercise1***

8. Lab 8 – Timers

DEMO: Exercise 1.

- For executing this program you must use the original bitstream, which is provided at:
  [RVfpgaPath]/RVfpga/src/rvfpganexys.bit
- PlatformIO project available at:

*[RVfpgaPath]/RVfpga/Labs/RVfpgaLabsSolutions/ProgramsAndDocuments/Lab08/**Exercise1***

DEMO: Exercise 3.

- For executing this program you must use the modified bitstream with support for PWM, which is provided at:
  [RVfpgaPath]/RVfpga/Labs/RVfpgaLabsSolutions/Modified_RVfpgaSystem/RVfpgaSystem_Labs6-10/src/rvfpganexys.bit
- Use the PlatformIO project available at:
  *[RVfpgaPath]/RVfpga/Labs/RVfpgaLabsSolutions/ProgramsAndDocuments/Lab08/**Exercise3***

9. Lab 9 – Interrupt-Driven I/O

In this demo and hands on exercise, use the original bitstream again, which is provided at:

[RVfpgaPath]/RVfpga/src/rvfpganexys.bit

DEMO: Example from Section 5.

- First show the execution of **LED-Switch_C-Lang** program. Use the PlatformIO project provided at:
  [RVfpgaPath]/RVfpga/Labs/Lab09/LED-Switch_C-Lang
- Then show the execution of **LED-Switch_7SegDispl_C-Lang** program. Use the PlatformIO project provided at:
  [RVfpgaPath]/RVfpga/Labs/Lab09/LED-Switch_7SegDispl_C-Lang
- Finally show the execution of **LED-Switch_7SegDispl_Interrupts_C-Lang** program. Use the PlatformIO project provided at:
  [RVfpgaPath]/RVfpga/Labs/Lab09/LED-Switch_7SegDispl_Interrupts_C-Lang

HANDS ON: Exercise 1. Test this exercise, which uses interrupts both for the switches and for the timer. Use the platformIO project available at:

*[RVfpgaPath]/RVfpga/Labs/RVfpgaLabsSolutions/ProgramsAndDocuments/Lab09/**Exercise1***

10. Lab 10 – Serial Buses

DEMO: Exercise 1.

- For executing this program you must use the original bitstream, which is provided at:
  [RVfpgaPath]/RVfpga/src/rvfpganexys.bit
- PlatformIO project available at:
  *[RVfpgaPath]/RVfpga/Labs/RVfpgaLabsSolutions/ProgramsAndDocuments/Lab10/**Accelerometer_7SegDisp***

DEMO: Exercise 3.

- For executing this program you must use the original bitstream, which is provided at:
  [RVfpgaPath]/RVfpga/src/rvfpganexys.bit
- Use the PlatformIO project available at:
  *[RVfpgaPath]/RVfpga/Labs/RVfpgaLabsSolutions/ProgramsAndDocuments/Lab10/**Scanf_Printf_7SegDispl***

PART 2. RVfpga Labs – Part 2 – Labs 11 to 20

11. Lab 11 – SweRV EH1 Configuration, Organization & Performance Monitoring

HANDS ON: Task from Section 2.D:

TASK: Replicate the simulation from Figure 11 (slide 146) and Figure 12 (slide 148) on your own computer by following these steps (as described in detail in Section 7 of the GSG):

- If necessary, generate the simulation binary (*Vrvfpgasim*). This step is not necessary in the Workshop !!
- In PlatformIO, open the project provided at: *[RVfpgaPath]/RVfpga/Labs/Lab11/ExampleProgram*.
- Establish the correct path to the RVfpga simulation binary (*Vrvfpgasim*) in file *platformio.ini*. Remember that binaries are provided at: *[RVfpgaPath]/RVfpga/verilatorSIM/OriginalBinaries*
- Generate the simulation trace with Verilator (Generate Trace).
- Open the trace using GTKWave.
- Use files *test_1.tcl* and *test_2.tcl* (provided at *[RVfpgaPath]/RVfpga/Labs/Lab11/ExampleProgram*) for opening the same signals as the ones shown in Figure 11 and Figure 12. For that purpose, on GTKWave, click on *File → Read Tcl Script File* and select the *test_1.tcl* or *test_2.tcl* file.
- Click on *Zoom In* () several times and move to 48600ps (or any other iteration of the loop, except the first one).

HANDS ON: Task from Section 3.B:

TASK: Execute the program from Figure 13 (slide 151), provided at *[RVfpgaPath]/RVfpga/Labs/Lab11/HwCounters_Example*, on the Nexys A7 board as explained in the GSG. You should obtain the results shown in Figure 14 for the four measured events. Explain and justify the results.
Measure other events in the Hardware Counters for the program from Figure 13.

12. Lab 12 – Arithmetic / Logic Instructions: The `add` Instruction

DEMO: Exercise 7:

7) (*The following exercise is based on exercise 4.4 of [PaHe] and exercise 1 of Chapter 7 of the textbook by S. Harris and D. Harris, "Digital Design and Computer Architecture: RISC-V Edition" [DDCARV].*)

When silicon chips are fabricated, defects in materials (e.g., silicon) and manufacturing errors can result in defective circuits. A very common defect is for one signal wire to get "broken" and always register a logical 0. This is often called a "stuck-at-0" fault. Determine the effect of each of the control bits included in signal `i0_ap` (a signal of type `alu_pkt_t`) being stuck at 0.

Do not perform the simulation, just show the solution, which is provided at:

[RVfpgaPath]/RVfpga/Labs/RVfpgaLabsSolutions/ProgramsAndDocuments/Lab12/Lab12_SolutionsTasksAndExercises

13. Lab 13 – Memory Instructions: `lw` and `sw` Instructions

No hands-on or demo.

14. Lab 14 – Structural Hazards

HANDS ON: Task from Section 2.A.

> **TASK:** Remove the `nop` instructions included within the loop from Figure 1 (shown in slide 182) and measure different events (cycles, instructions/multiplies committed, etc.) using the Performance Counters available in SweRV EH1, as explained in Lab 11. Is the number of cycles as expected after analysing the simulation from Figure 2 (shown in slide 183)? Justify your answer.
> Now reorder the code within the loop trying to reach the ideal throughput. Justify the results obtained in the original code and in the reordered one.

- The program is provided at:
  [RVfpgaPath]/RVfpga/Labs/Lab14/MUL_Instruction

15. Lab 15 – Data Hazards

DEMO: Task from Section 2.A.

> **TASK:** Remove all `nop` instructions in the example from Figure 2 (shown in slide 195). Draw a figure similar to Figure 3 (shown in slide 196) for two consecutive iterations of the loop, then analyse and confirm that the figure is correct by comparing it to a Verilator simulation, and finally compute the IPC by using the Performance Counters while executing the program on the board.

- Do not perform the simulation, just show the solution, which is provided at:
  [RVfpgaPath]/RVfpga/Labs/RVfpgaLabsSolutions/ProgramsAndDocuments/Lab15/Lab15_SolutionsTasksAndExercises

DEMO: Task from Section 3.

> **TASK:** Disable the Secondary ALU as explained in Lab 11 and analyse the example from Figure 11 (slide 203) both with a Verilator simulation and with an execution on the board.

- Do not perform the simulation, just show the solution for this task and the previous one (where the Secondary ALU is enabled), which are provided at:
  [RVfpgaPath]/RVfpga/Labs/RVfpgaLabsSolutions/ProgramsAndDocuments/Lab15/Lab15_SolutionsTasksAndExercises

16. Lab 16 – Control Hazards & Branch Instructions

HANDS ON: Task from Section 3.

> **TASK:** In the example from Figure 2 (slide 212), remove all the `nop` instructions and analyse the simulation. Then compute the IPC with the Performance Counters by executing the program on the board.
>
> Enable the branch predictor used in SweRV EH1 (by commenting out the two initial instructions in Figure 2) and analyse the simulation and the execution on the board.
>
> Compare the two experiments and explain the results.

- The program is provided at:
  [RVfpgaPath]/RVfpga/Labs/Lab16/BEQ_Instruction

- Do not perform the simulations, just show the waveforms, but do execute on the board for measuring the IPC:
  [RVfpgaPath]/RVfpga/Labs/RVfpgaLabsSolutions/ProgramsAndDocuments/Lab16/Lab16_SolutionsTasksAndExercises

17. Lab 17 – Superscalar Execution

DEMOs: Exercises 3, 5 and 6.

- Show the solutions, which are provided at:
  [RVfpgaPath]/RVfpga/Labs/RVfpgaLabsSolutions/ProgramsAndDocuments/Lab17/Lab17_SolutionsTasksAndExercises

18. Lab 18 – Adding New Features

HANDS ON: Exercises 1, 3 and 4.

Exercise 1:

- Read the first part of the exercise statement to understand the process of including a new instruction in the SweRV EH1 core.
- Note that you don't need to follow the process explained, as the bitstream of the processor including the new instructions is provided in the solutions.

Exercise 3:

- Read the exercise statement. Do not perform the simulations.
- Then, analyse and test on the board the program from:
  [RVfpgaPath]/RVfpga/Labs/RVfpgaLabsSolutions/Modified_RVfpgaSystem/RVfpgaSystem_Lab18/FloatingPoint/DotProduct_Comparision

- For that purpose, you can use the bitstream provided at: *[RVfpgaPath]/RVfpga/Labs/RVfpgaLabsSolutions/Modified_RVfpgaSystem/RVfpgaSystem_Lab18/FloatingPoint/src*
- Compare the results of the dot product computed in software and in hardware: ***sum_swemul*** vs. ***sum_hwimpl***.
- Compare the number of cycles for the execution in software and in hardware.

Exercise 4:
- Read the exercise statement.
- Analyse and test on the board the program from: *[RVfpgaPath]/RVfpga/Labs/RVfpgaLabsSolutions/Modified_RVfpgaSystem/RVfpgaSystem_Lab18/FloatingPoint/BisectionAlgorithm*
- For that purpose, you can use the same bitstream as in Exercise 3, which is available at: *[RVfpgaPath]/RVfpga/Labs/RVfpgaLabsSolutions/Modified_RVfpgaSystem/RVfpgaSystem_Lab18/FloatingPoint/src*
- Compare the results of the dot product computed in software and in hardware: ***zero_swemul*** vs. ***zero_hwemul***.
- Compare the number of cycles for the execution in software and in hardware.

19. Lab 19 – Instruction Cache

No hands-on or demo.

20. Lab 20 – ICCM, DCCM & Benchmarking

HANDS ON: Section 3
- **Section 3.A**: No compiler optimizations or DCCM/ICCM
- **Section 3.B**: Using the DCCM
- **Section 3.C**: Using the DCCM and compiler optimizations