Design of an Academic RISC-V Core

Bachelor's Degree Final Project (TFG)

Pau Casacuberta, Raimon Casanova, Ricardo Martínez, David Castells and Lluís Terés

Universitat Autònoma de Barcelona
escola d'enginyeria

RED2018-102384-T

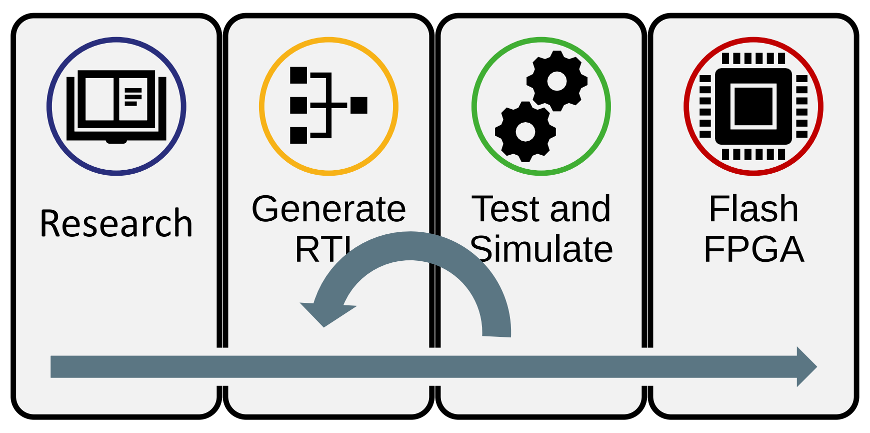
Motivation

- Update academic material.
- Gain experience in HW design.
- Introduction to the RISC-V ISA.
- Explore Open Source toolchains for HW design.

Objectives

- Design a RISC-V core using the **RV32I** ISA.
- Development of a basic modelling & verification environment written in Verilog.
- Implementation & Test in an FPGA.
- Adapt the design to work in Pulpino platform.

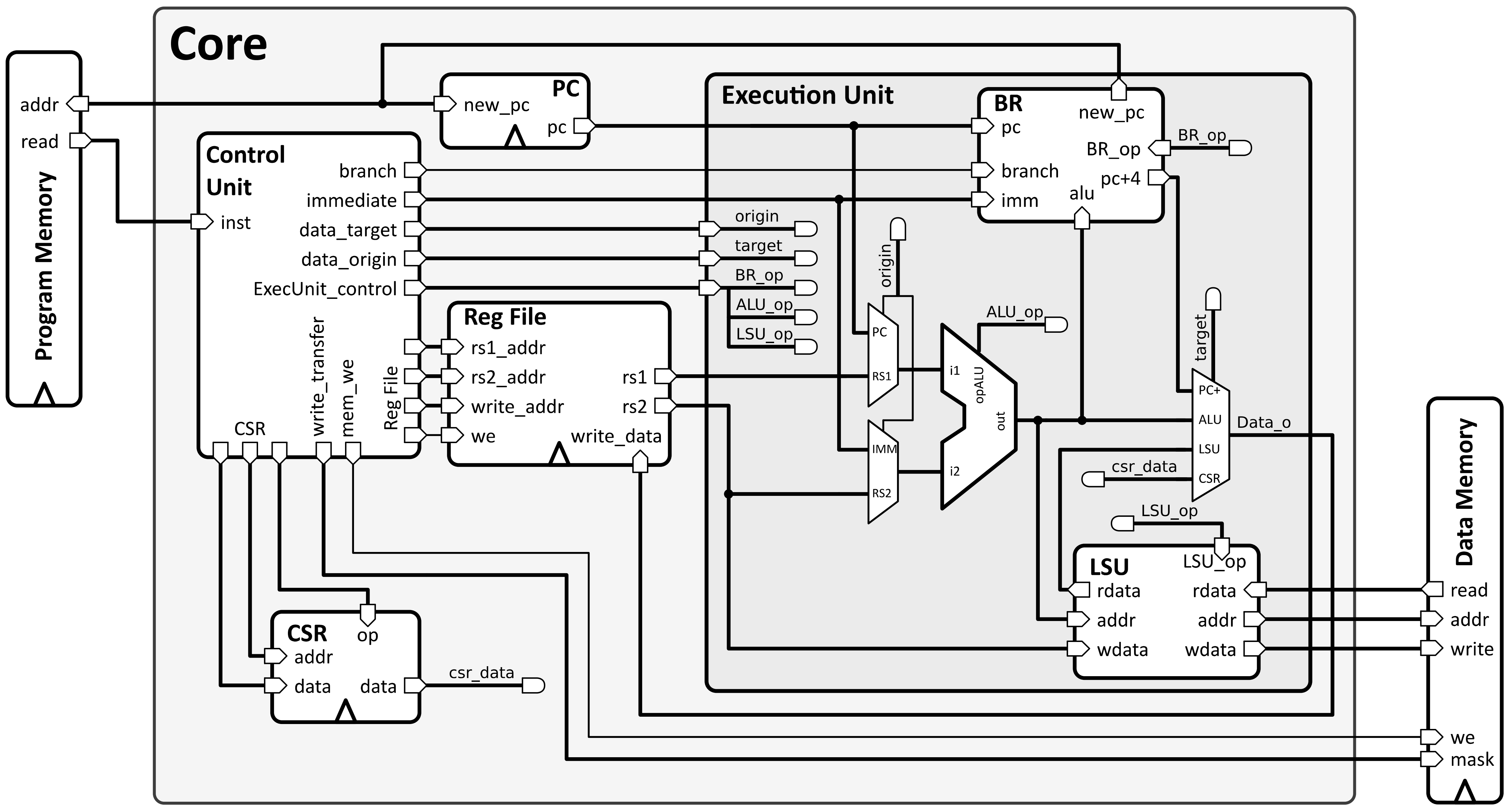
Methodology



Architecture

Simple Design:

- Single Cycle Processor.
- No program or data memory cache.
- No Branch Prediction mechanism.
- No interruptions.
- User Level CSR (only 1).

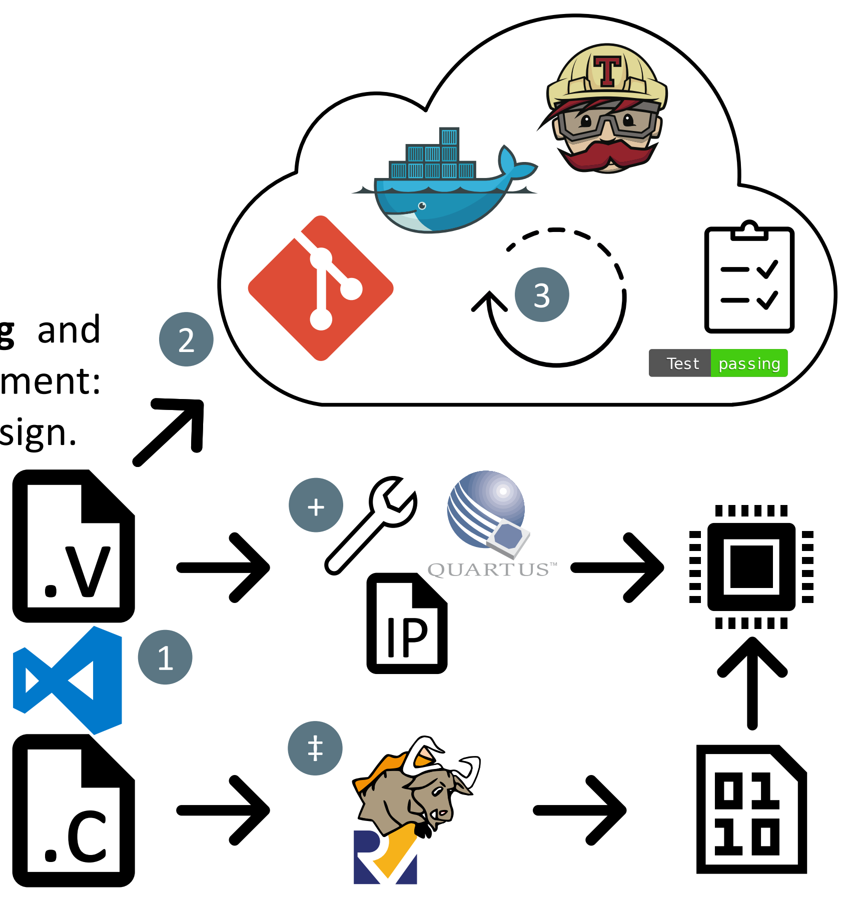


Design Workflow

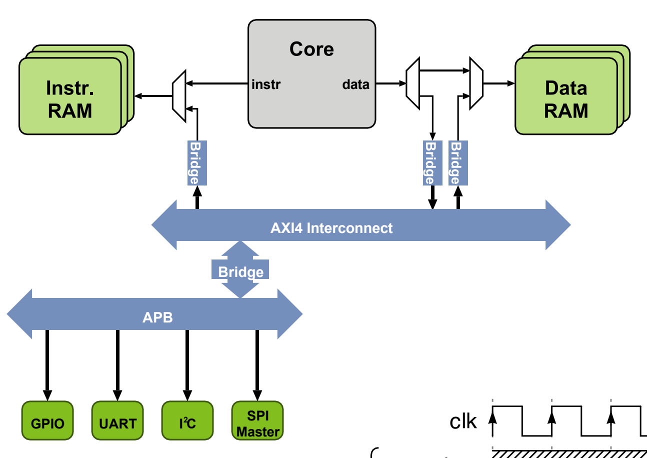
1. Write Verilog HDL code with a text editor (**VSC**).
2. Update a **Git** repository.
3. Using Containers (**Docker**) with **Icarus Verilog** and Continuous Integration (**Travis**) as a test environment: a reference model is compared with the new design.

+ **Quartus II** suite is used to port the design into the FPGA using **IP libraries** to control other elements of the development board.

‡ To generate binaries from C code we need to compile C code with the **gnu-riscv-toolchain** and then load it to the FPGA.

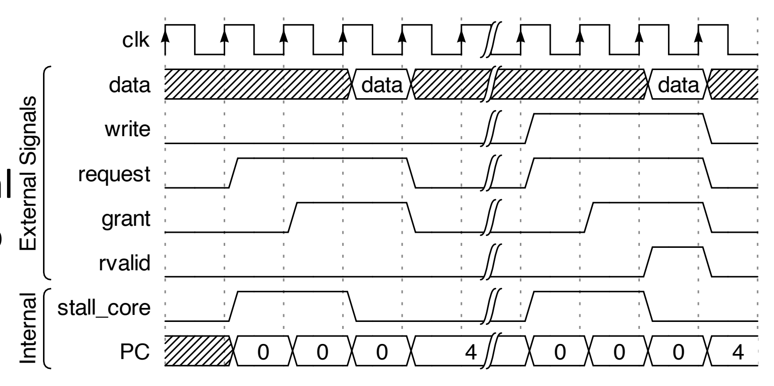


Adaptation to Pulpino



Memory is shared by different peripherals. A Request/acknowledge protocol between the core and the memory is necessary.

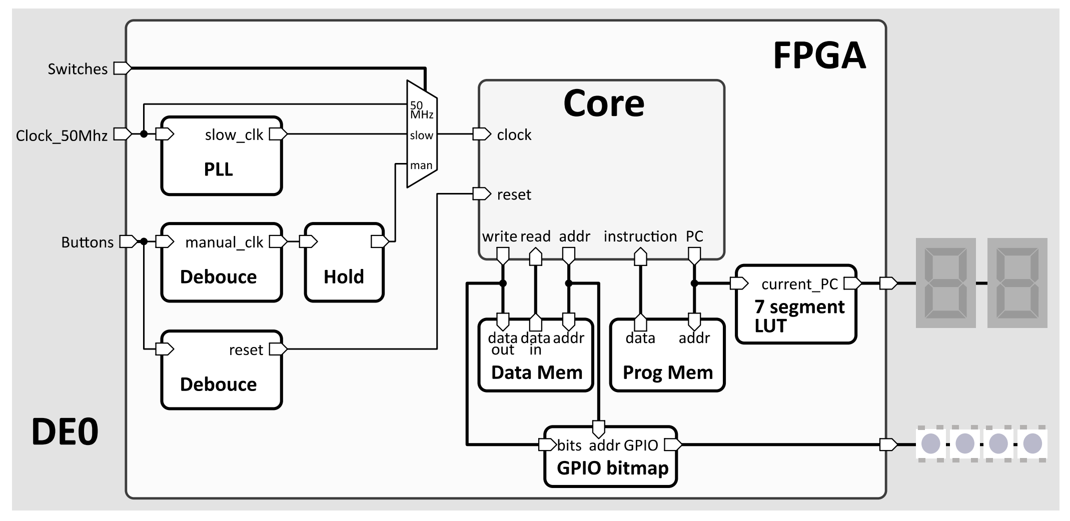
To implement the protocol additional signals are used to access the memory.



The core place a **request** to perform load/store operations to the data memory. The core is **stalled** until the access is **granted**.

FPGA

The core has been implemented in a **DE0** development board from TeraAsic. Several auxiliary modules have been added for the implementation: a **PLL** to slow down the **clock**, **debouncers** to use **buttons**, **Memories** generated from **Altera IPs**, **LUTs** to display the **PC** value with **7 segment displays** and a module to bit map a memory word to some **LEDs** in the board.



Conclusion

Main objectives achieved: the design can be simulated in a Verilog compiler and execute RV32I instructions and can run compiled C programs on an FPGA.

"The process of creating a core from the ground up is very helpful to understand how computing works, and implementing it in an FPGA teaches the hazards of real Hardware."

The source code of this project is available on GitHub: https://github.com/4a1c0/RV32i-Verilog

"The standardization of an ISA is a great way to allow more and better accessible development tools, and RISC-V is perfect for that."