Learning Computer Architecture with a Visual Simulation of RISC-V Processors

Esteban Stafford*, Borja Pérez†, and Jose Luis Bosque‡

Department of Computer Science and Electronics Engineering
Universidad de Cantabria

Abstract

Teaching computer architecture is challenging due to its abstract concepts and complexity. Visual learning tools help reinforce key ideas but often lack flexibility for direct architectural modifications. This work presents an enhanced visual simulation approach using Logisim Evolution, allowing students to implement and debug single-cycle and pipelined RISC-V processors. New programmable components enable students to modify functionality dynamically, improving comprehension. The methodology has been successfully integrated into computer architecture courses, demonstrating effectiveness in laboratory assignments and exams. The results indicate that interactive visual tools significantly enhance learning outcomes for students and teaching efficiency for instructors.

Introduction

Learning computer architecture is a challenging endeavour due to its abstract and complex nature. Traditional teaching methods often struggle to convey the intricate interactions between hardware components and instruction execution. Visual learning tools play a crucial role in reinforcing these concepts by providing intuitive representations of processor operations. However, existing visual simulators offer limited flexibility, as they are primarily configuration-driven, restricting students' ability to implement and modify architectural components.

Objective

The aim of this work is to select and adapt a visual digital simulator that allows students to implement both single-cycle and pipelined RISC-V processors [1, 2]. This approach enables students to construct parts of the architecture, diagnose issues, and debug their designs, fostering a deeper understanding of computer architecture.

Selected Simulator and Enhancements

Logisim Evolution was chosen as the simulation platform due to its feature set and portability, being implemented in Java [3]. However, some enhancements were necessary to support the desired learning outcomes. Additional components such as the ALU and register file were introduced. More significantly, a new programmable family of components was developed. These components are defined like standard Logisim components, with inputs, outputs, and behaviour specified in Java. However, in this case, students can modify the behaviour dynamically. By clicking on "Edit Contents," a code editor appears, allowing students to write code to implement the functionality of complex components without requiring detailed knowledge of digital electronics.

Course Integration and Laboratory Assignments

These enhancements have been used as the building blocks for the lab assignments currently employed in two computer architecture courses for computer science and telecommunications students. The goal of these lab assignments is to help students better understand how processors work by having them implement parts of their functionality. The assignments covert two processor architectures: single-cycle and pipeline.

Single-cycle Processor

Students are provided with a schematic containing the clock, program counter, and instruction memory. Their task is to integrate various processor components to enable execution of basic instructions such as `add`, `sub`, `and`, `or`, `lw`, `sw`, and `beq`. The final design is shown in Figure 1. Additionally, they must implement new instructions, such as a load with an accumulator, which adds the memory value to the current register contents.

*esteban.stafford@unican.es
†borja.perez@unican.es
‡joseluis.bosque@unican.es

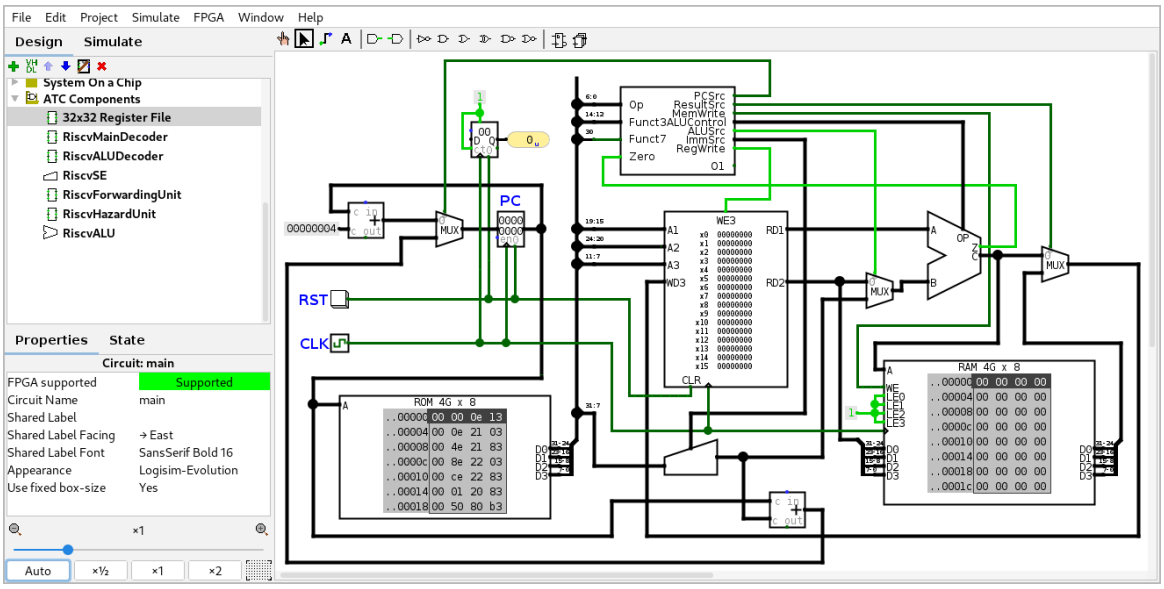

Figure 1: *Single-cycle RiscV Processor in Logisim Evolution*

Pipelined Processor

Students receive a pre-built pipelined RISC-V processor and a program exhibiting data hazards. Initially, they must manually insert `nop` instructions to ensure correct execution. The challenge then progresses to reducing the number of inserted `nop` instructions. This involves:

- Advancing branch execution to the ID stage by adding a comparator.
- Implementing a forwarding unit, requiring students to integrate multiplexers and modify the forwarding logic.
- Adding a hazard detection unit to handle load dependencies.

Certain components, such as the forwarding unit, are editable, allowing students to program their behaviour directly within Logisim Evolution.

Results and Experience

These laboratory assignments have been used successfully in classrooms for several years. Since they were initially designed for MIPS and later adapted to ARM, the transition to RISC-V has been smooth. The interactive nature of the simulator even allows for exam exercises where students must identify implementation errors or explain specific processor behaviours. The RiscV versions of the assignments have been well received by students, who appreciate the hands-on experience and the ability to modify the processor components. Each assignment is designed to be completed in approximately four hours. This is the first year that the RISC-V version of the assignments has been used with a class of 70 students, divided into groups of 20 maximum.

Conclusion

The experience demonstrates that interactive learning tools significantly enhance both student comprehension and instructor effectiveness. The ability to modify and debug processor components provides valuable hands-on experience, reinforcing key concepts in computer architecture.

Acknowledgements

With support from the Spanish Science and Technology Commission under contract PID2022-136454NB-C21, the Ministerio de Ciencia e Innovación; Proyectos de Transición Ecológica y Digital 2021 under grant TED2021-131176B-I00 and the European HiPEAC Network of Excellence.

References

[1] Sarah Harris and David Harris. *Digital Design and Computer Architecture, RISC-V Edition*. Morgan Kaufmann, 2013. ISBN: 978-0128200643.

[2] David Patterson and Andrew Waterman. *The RISC-V Reader: An Open Architecture Atlas*. Strawberry Canyon, 2017. ISBN: 0999249118.

[3] Logisim Evolution Team. *Logisim Evolution*.