Aperture - Computer Architecture Made Visual

Christofer Rodrigues[1], **Rogério Aparecido Gonçalves**[1] **e João Fabrício Filho**[1]

[1]Departamento Acadêmico de Computação (DACOM)
Universidade Tecnológica Federal do Paraná (UTFPR) – Campus Campo Mourão

christofer_daniel12@hotmail.com,{rogerioag,joaof}@utfpr.edu.br

Abstract. *Computer Architecture and Organization (CAO) is a recurring curricular component in technology courses. However, the content covered in this subject ends up being limited to abstract concepts about how components work, sometimes aided by the use of diagrams. Thus, we propose the implementation of a material focused on visualization and interactivity with the components studied, allowing the observation of the signals that pass through the circuits, such as in an adder, or the control signals of a processor. The material produced incrementally delivers basic digital logic circuits, such as adders, or memories, and gradually increases in complexity, reaching the implementation of a MIPS processor with a 5-stage pipeline based on the book by David A. Patterson and John LeRoy Hennessy. With this work, we intend to facilitate the process of studying AOC by providing material with a focus on visualization and interactivity that complements traditional forms of teaching.*

1. Introduction

Computer Architecture and Organization (CAO) is a common discipline of computer science course curriculums. However, the contents covered in this subject can suffer from impediments, such as the difficulty of visualizing the constituent components of a computer processor without specialized equipment, a situation caused by the physical characteristics of modern electronic components.

This situation leads to the presentation of this content being limited to the behavior of the components, or to the use of conceptual diagrams. Thus, these concepts become abstract when approached in the classroom, which makes it difficult to assimilate the content and requires more mental effort from students to understand the concepts and how they connect.

Various areas of science have explored the use of simulators to facilitate the understanding of abstract content. For example, the PHeT platform[1], which aggregates a variety of interactive simulators for areas such as physics and chemistry. For AOC, there are simulators such as MARS [Vollmar and Sanderson 2006] and EDUMips [Patti et al. 2012]. However, these implementations lack one feature or another, such as not providing visualization of the circuit executing the code, or the lack of interactivity and the ability to modify the circuit. Furthermore, open implementations increase the possibilities for adaptations and modifications for educational and research use [Carro 2021].

To deal with these problems, and based on previous results [Rodrigues et al. 2024], we present an Open Educational Resource (OER)

[1]https://phet.colorado.edu/

with a focus on visualization and interactivity with the simulated content. This is achieved by using a digital circuit simulator called logisim-evolution [2]. It was chosen for its ease of use and ability to interact with the circuits, as well as the possibility of visualizing the operation of these circuits, showing the high or low states of the wires, or the contents of registers or wires that pass multiple bit values.

The OER in this work covers content ranging from the implementation of logic gates using transistors to the construction of a MIPS data path with a five-stage pipeline, mostly inspired by the descriptions in the book Patterson and Hennessy [Patterson and Hennessy 2013]. All the implementations and contents of the material are available on the GitHub repository[3] and also on the project's landing page[4].

The main feature of this work is its breadth of content, ranging from the exploration of digital logic concepts to content covered in AOC. And all the material being interactive, with the user being able to explore its workings by modifying inputs and outputs, and even altering the circuits completely, if they so wish. The comprehensiveness of the content allows the user to explore how each component of the architecture works, eliminating "black box" effects in understanding, and the interactivity allows exploration and experimentation, which can lead to a greater understanding of how the circuits work.

2. Objectives

In the study carried out in [Fernandes and Silva 2017], it was possible to observe an improvement in the grades obtained by the students in the AOC and Operating Systems subjects. In addition, the students reported that the use of a simulator made learning AOC easier and increased their motivation in the subjects.

Based on this, we intend to collaborate with the state of the art by bringing material focused on visualization and interactivity with the circuit that can be used in conjunction with traditional classes. This situation touches on concepts such as Dual Coding Theory, which explains how the brain uses verbal and non-verbal channels to process and store information [Clark and Paivio 1991].

The material covers fundamental knowledge of digital circuits, such as the construction of logic gates, as well as more complex circuits, and progresses towards the implementation of a MIPS processor with a five-stage pipeline. Starting from basic concepts and increasing the degree of complexity of the implementations has foundations on the theory known as Elaborative Encoding, which describes that memories are better stored when connected to previously known content [Bradshaw and Anderson 1982].

With the material in the format presented, students can explore the workings of the various components, such as memory, registers, or muxes, individually, as if they were opening "black boxes" whenever they open the circuit file that implements each of them. In this way, we hope that students will better retain the content they have studied and be able to use this knowledge in different areas of computing.

[2]https://github.com/logisim-evolution/logisim-evolution

[3]`https://github.com/ChristoferLv/Aperture-Releases`

[4]`https://christoferlv.github.io/ProjetoAperture/`

3. Methodology

The material was implemented in the logisim-evolution software, which is a digital circuit simulator. It allows you to build circuits by dragging and dropping components that are natively available in the software, such as logic gates, multiplexers, and registers. Although the software allows the use of customized components implemented in Java, the material in this work uses only the native components, in order to avoid compatibility problems and make it easier to use.

The material provided is organized in the form of various circuit files made in Logisim that cover some specific content. For example, in the materials on digital circuits, you can find circuits on arithmetics, and when this file is loaded into Logisim, you can find the sub-circuits that implement each of the arithmetic operations. The circuits are all available on the GitHub repository, or on the project's landing page, designed to facilitate access to the resources, which also has a gallery tab with images of each of the circuits.

The implementations and design of the MIPS processor built at the end of the material trail are strongly based on the descriptions found in the book by Patterson and Hennessy [Patterson and Hennessy 2013]. This choice is made because the authors were primarily responsible for formalizing the dissemination of the RISC philosophy and the creation of MIPS processors.

4. Results

From this process, the results obtained are an OER, available on the GitHub repository, which can be used to help teach AOC and other digital circuit concepts. The material focuses on visualization and interactivity with the content being studied, supported by the capabilities of the simulator used to build the circuits.

The material consists of two main groups, basic digital circuit components and implementations of the MIPS datapath. The first group contains circuits that explore eight different concepts: logic gates, logic equivalences using NAND and NOR, encoders, multiplexers, flip-flops, registers, arithmetic, containing implementations for the 4 basic arithmetic operations and static memories (SRAM).

The group of MIPS datapath implementations includes "partial datapaths" and "complete datapaths". The "partial datapaths" have no control unit and are made up of seven implementations, six of which are a circuit sufficient to perform the add, addi, beq, j, lw and sw operations separately, in which the student can execute code with these instructions, while observing the resources required to implement them. As these are incremental implementations, the last implementation unites all the other 6, but still without a control unit, so it is up to the user to turn on the control signals correctly so that they can explore and understand the role of this component.

In the "complete datapaths" section, there are eight implementations, one of which is a single-cycle MIPS datapath with a set of twenty instructions, specified in the documentation on the project's GitHub, and the other seven implementations apply the pipeline technique. These progress gradually, and in different combinations, exploring the different techniques for improving datapath performance, according to the specifications described in [Patterson and Hennessy 2013].

With the material developed, a teacher can use it to complement the process of teaching digital logic or computer architecture as supplementary material, used in conjunction with class explanations, showing how the components work as soon as the content is explained. Another possibility is to give students activities that encourage them to explore the material's visual and interactive capabilities. For example, ask students to try out some of the circuits and try to understand their behavior before or during a lesson.

5. Conclusion

This work presents an OER that covers the fundamentals of digital circuits and computer architecture. In the material, various traditional digital circuit components are implemented, and the degree of complexity increases until it reaches the implementation of a MIPS processor with a 5-stage pipeline with various performance gain techniques, following the specifications of the Patterson and Hennessy's book.

The circuits implemented in the material prioritize the understanding rather than any optimization that can be done to save components. This material can be used freely by teacher or students as an additional resource that brings a visual and interactive component to facilitate the understanding of the concepts covered in classes. In the future, we intend to evaluate with students the effectiveness of using the material.

In future work, the material will implement the RISC-V ISA, that will be available alongside the current one. RISC-V adopts open hardware standards that boost multiple possibilities for education, allowing for different versions of a CPU project. In addition, the open hardware attracts government and enterprise interests in the RISC-V standards.

References

Bradshaw, G. L. and Anderson, J. R. (1982). Elaborative encoding as an explanation of levels of processing. *Journal of Verbal Learning and Verbal Behavior*, 21(2):165–174.

Carro, L. (2021). Hardware aberto, uma análise de possibilidades. *Computação Brasil*, 46(46):29–32.

Clark, J. and Paivio, A. (1991). Dual coding theory and education. *Educational Psychology Review*, 3:149–210.

Fernandes, S. and Silva, I. (2017). Relato de experiência interdisciplinar usando mips. *International Journal of Computer Architecture Education*, 6:52–61.

Patterson, D. A. and Hennessy, J. L. (2013). *Computer Organization and Design, Fifth Edition: The Hardware/Software Interface*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 5th edition.

Patti, D., Spadaccini, A., Palesi, M., Fazzino, F., and Catania, V. (2012). Supporting Undergraduate Computer Architecture Students Using a Visual MIPS64 CPU Simulator. *IEEE Transactions on Education*, 55(3):406–411.

Rodrigues, C., Gonçalves, R. A., and Fabrício Filho, J. (2024). Mips processor implemented in a visual simulator for educational use. *International Journal of Computer Architecture Education*, 13(1):33–42.

Vollmar, K. and Sanderson, P. (2006). Mars: an education-oriented mips assembly language simulator. *SIGCSE Bull.*, 38(1):239–243.