ASEE AMERICAN SOCIETY FOR ENGINEERING EDUCATION

Paper ID #44617

Enhancing Student Understanding of Digital logic and Computer Architecture Through Turing Complete Game Challenges

Eric McKanna, Ohio Northern University
Dr. Firas Hassan, Ohio Northern University

Firas Hassan is an associate professor at Ohio Northern University. He got his Ph.D. from The university of Akron. His research interest are in the area of embedded computing of real-time image processing techniques.

Enhancing Student Understanding of Digital Logic and Computer Architecture Through Turing Complete Game Challenges

Eric McKanna, Dr. Firas Hassan
Department of Electrical & Computer Engineering & Computer Science
Ohio Northern University
Ada, Ohio 45810
Email: e-mckanna@onu.edu

Abstract

Turing Complete[1] is a game released on the Steam[2] platform designed to teach digital logic and computer architecture concepts through a series of challenges and problems. Its skill-tree approach covers digital logic, binary arithmetic, and memory. The end goal is to create a turing complete computer through building blocks developed during each different section of the tree. In this paper, we suggest adapting and modifying problems from the game to emphasize key comprehension points in the students' coursework to both gauge understanding and build a more dynamic skill set with logic gates. In addition to being a superb source of content for labs or homework assignments, the game also provides students with an excellent sandbox and simulation tool to develop and experiment with their ideas.

Introduction

The incorporation of gamification into education has garnered substantial attention in recent years.[3] A wealth of research has focused on evaluating the effectiveness of games as learning tools. A comprehensive meta-analysis of this body of research uncovered "significant, positive effects of gamification on cognitive, motivational, and behavioral learning outcomes."[4] This significance is particularly noteworthy in higher education, where "undergraduates thrive with gamification."[5] Given the clear benefits of gamification, and having identified a game that aligns well with our curriculum, we propose its integration into Digital Logic and Computer Architecture courses.

This paper aims to propose a method to enhance students' comprehension of digital logic and computer architecture through engaging challenges within the Turing Complete game. Specifically, the paper will explore a challenge involving the design of a decoder in the game, wherein students could be tasked with creating a circuit that produces a true output when a minimum of 2 out of 4 input bits are set to high. Initially introduced during the study of Karnaugh maps (Kmaps), this challenge could evolve throughout the semester into a more complex task: constructing a decoder that yields a true output for n high bits out of a total of 32 input bits. To successfully tackle this advanced problem, students would be required to leverage higher-level components such as shift registers, accumulators, and comparators, which are thoroughly explained in later segments of the course curriculum.

I myself used the game when taking Digital Logic and Computer Architecture and found it to be an invaluable addition to my coursework. While other students were struggling to understand the concepts, I was already working toward designing my own processor. I was able to experiment

with ideas, test implementations, and customize the hands-on experience I find necessary to learn new material. Not only that, but I was enjoying myself! After taking Computer Architecture, I approached my professor about exploring incorporating the game into the coursework at my university as a means to assist struggling students through alternative learning methods. This paper is a formalized version of the proposal that I made to my professor.

In addition to developing unique projects for students, this paper suggests an innovative approach to easing the transition between digital logic and computer architecture. We propose tasking students with independently constructing an 8-bit processor in the one-year interval between these two courses. Success in this endeavor not only provides a sense of accomplishment but also offers the opportunity for extra credit to be offered in the subsequent computer architecture class. Furthermore, the students would be ready to face the difficulties of computer architecture head-on.

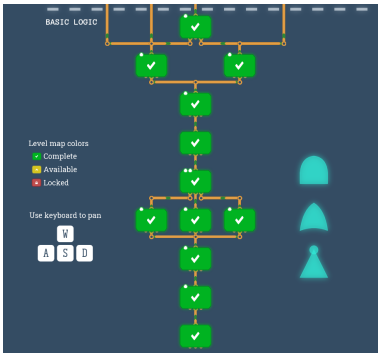
In addition to the standard challenges and projects found in the game, this paper delves into the transformative potential of the Turing Complete game's sandbox and simulation features. These features serve as powerful tools to enhance student comprehension in labs and homework assignments. The game's sandbox provides a virtual space for students to experiment, simulate, and visualize digital logic concepts, offering a dynamic alternative to traditional tools like ModelSim or Vivado. By incorporating the game's features into their coursework, students can gain a more interactive and hands-on understanding of complex topics, thereby improving their overall learning experience in both lab settings and individual assignments.

The Turing Complete Game

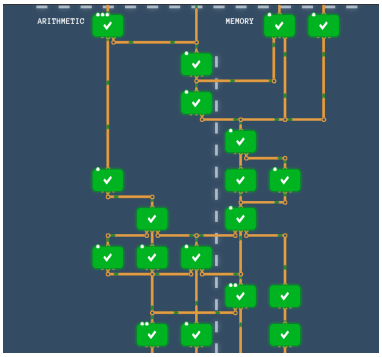
In the course of our Computer Scientist and Computer Engineering tracks, students encounter two, unique and challenging courses: Digital Logic and Computer Architecture. Much of the complexity is derived from difficulties in conceptualizing how components interact and change from one state to the next. In the past, this was addressed through several simulations done in both Vivado and ModelSim during labs. This allows the students to have a visual and tangible experimentation ground in which to develop mental frameworks of understanding. Unfortunately, these tools, along with similar ones, can be unnecessarily complex, filled with distracting features. Additionally, the hardware is designed using VHDL, a language known for its complexity and obscurity. In response to these difficulties, we have introduced a unique solution: playing with a video game named "Turing Complete." Despite its status as a game, it is not simply there for entertainment. Instead, it serves as a guided sandbox where students can implement hardware using GUI components and observe real-time changes based on inputs.

Turing Complete aims to be a tool that engenders a deeper understanding of topics like logic gates, components, computer architecture, and assembly language. It does this through the gamification of standard digital logic and computer architecture curriculum elements via a skill tree that builds up into a significant achievement, the development of an 8-bit turing complete processor. Each leaf of the tree is a separate puzzle that falls within different requisite areas of knowledge necessary to build the components used in a processor. These areas of knowledge are delegated to sections of the skill tree with easy-to-understand labels such as arithmetic, memory, or logic gates. Each puzzle directly relies on knowledge or components developed in previous areas of the tree. In this

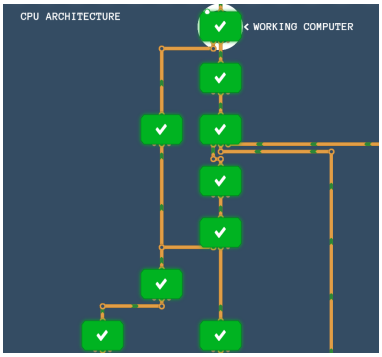
way, a player can quickly develop an intuitive and familiar expertise with various logical structures and patterns that are necessary for both coursework and a future career.



(a) Basic Logic Overview Section



(b) Arithmetic and Memory Sections



(c) Computer Architecture Section

Figure 1: Main Skill Tree

Each of these computational challenges, akin to puzzles, not only represents a creative undertaking but also catalyzes contemplating the scalability of a solution. For students, engaging with these challenges fosters cognitive growth, providing an avenue for honing problem-solving skills. Simultaneously, these challenges can potentially be leveraged as material from which to derive examination questions or laboratory exercises. The simplicity of the game's problems may obscure foundational underlying principles, making them ideal candidates for pedagogical exploration. Specifically, challenges would theoretically be selected by their ability to illustrate foundational principles such as scalability, efficiency, cost, and modularity. In the context of this research, we have selected a representative puzzle to demonstrate how the principle of scalability could be explored.

In addition to a solid set of problems and puzzles for a student to progress through, the game features extremely helpful simulation and labeling tools that allow students to perform real-time experiments and debugging, all without needing to write a single line of a hardware description language. Figure 2 shows how wire has a flowing color depending on whether it is high (green) or low (red). For colorblind users, the game also has moving segments that pass through the active wires, while the inactive (low) wires are static. All components operate on the same clock, and the game provides a clock counter that students can increment to observe changes based on dynamic inputs. In the component development sandbox, the game also provides a delay score, a gate score, and a critical path highlight.

Decoder Puzzle

Figure 3a is the level log for one of the challenges in the arithmetic section. The puzzle itself is a simple one, and it serves as a perfect illustration of both the power of the game's interface and its potential to spark curiosity. In our digital logic curriculum, students are taught how to use truth tables (Figure 3b), k-maps (Figure 3c), and sum of product expressions (Figure 3d) to solve problems such as this one. This puzzle, however, can usually be solved intuitively by a

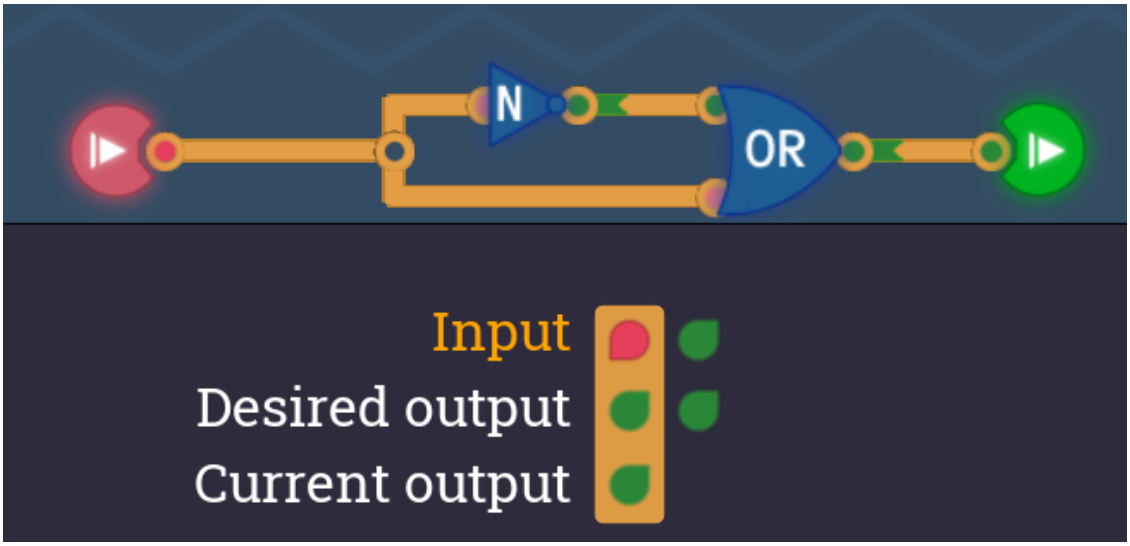
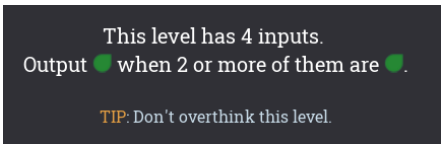


Figure 2: Live Simulation Features

little experimentation and simulation in the game. Later, the classwork can come in and provide a methodological explanation for the intuitive understanding while simultaneously providing the student with tools to solve similar, less obvious problems, in the future.

After students reach this challenge in the game, the instructor can present a theoretical problem, probing the scalability of the solution, which can be seen in Figure 4. Specifically, the instructor may introduce a hypothetical scenario where the number of inputs and required high bits can be arbitrarily increased, illustrating the exponential complexity of the problem. This complexity is especially evident when applying the methodological approach with truth tables and K-maps. Notably, for a scenario involving 'n' high bits out of thirty-two input bits, the truth table would increase to 2^{32} rows, rendering it impractical for manual processing. Such a solution would also be infinitely too large for any rational physical constraints. This question naturally prompts discourse on the limitations of static methods of problem-solving and segues into a discussion about more dynamic thought processes and solutions. These dynamic solutions involve the integration of more complex components introduced in later segments of the digital logic course, highlighting the necessity for sophisticated approaches to handle challenges of increasing scale and complexity.

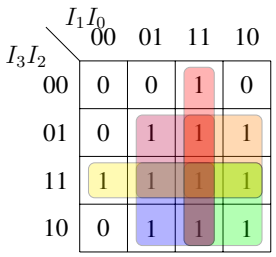
Once the material necessary for solving the scalability problem has been introduced, students may receive assignments around the problem in either a laboratory setting or as a take-home project. Evaluation of solutions encompasses considerations of correctness and efficiency, with an optional emphasis on optimizing resource utilization. Additionally, students are prompted to approach the problem creatively, fostering skill development even in the exploration of potentially incorrect solutions. This process not only serves as an evaluative measure but also cultivates valuable competencies applicable in professional environments.



(a) Decoder Puzzle Level Log

I_3	I_2	I_1	I_0	Out
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

(b) Truth table for the given logic circuit.



(c) Karnaugh Map

$$Out = I_3I_2 + I_1I_0 + I_2I_0 + I_2I_1 + I_3I_0 + I_3I_1 \quad (1)$$

(d) Sum of Products

Figure 3: Decoder Puzzle

A Scalable Solution

Our refined solution is composed of a shift register, an accumulator, and a comparator. The shift register facilitates the sequential feeding of the 32-bit input into the accumulator, with each clock cycle introducing a single bit. The accumulator incrementally adds one upon detecting a high bit until all the bits have been read. Subsequently, the input parameter 'n' is compared with the accumulated value. If 'n' is less than or equal to the stored value, the output signals high. This solution aptly demonstrates proficiency in a spectrum of fundamental skills acquired throughout our Digital Logic curriculum, encompassing Gates, Arithmetic, Flip-Flops, Shift Registers, and Clocked systems.

Significantly, all the requisite components for constructing the final version are conveniently available in the sandbox or can be implemented in separate sandbox instances. The user-implemented components, equally accessible, provide the flexibility to craft a refined and elegant final version. This approach not only showcases the applicability of learned skills but also emphasizes the versatility and adaptability of the ideas and material encountered in the lectures.

Introducing Hardware Optimization

Throughout the entire schematic depicted in Figure 5, various optimization techniques are at play. Positioned in the top left, the shift register serves to store the input, while the accumulator, situated

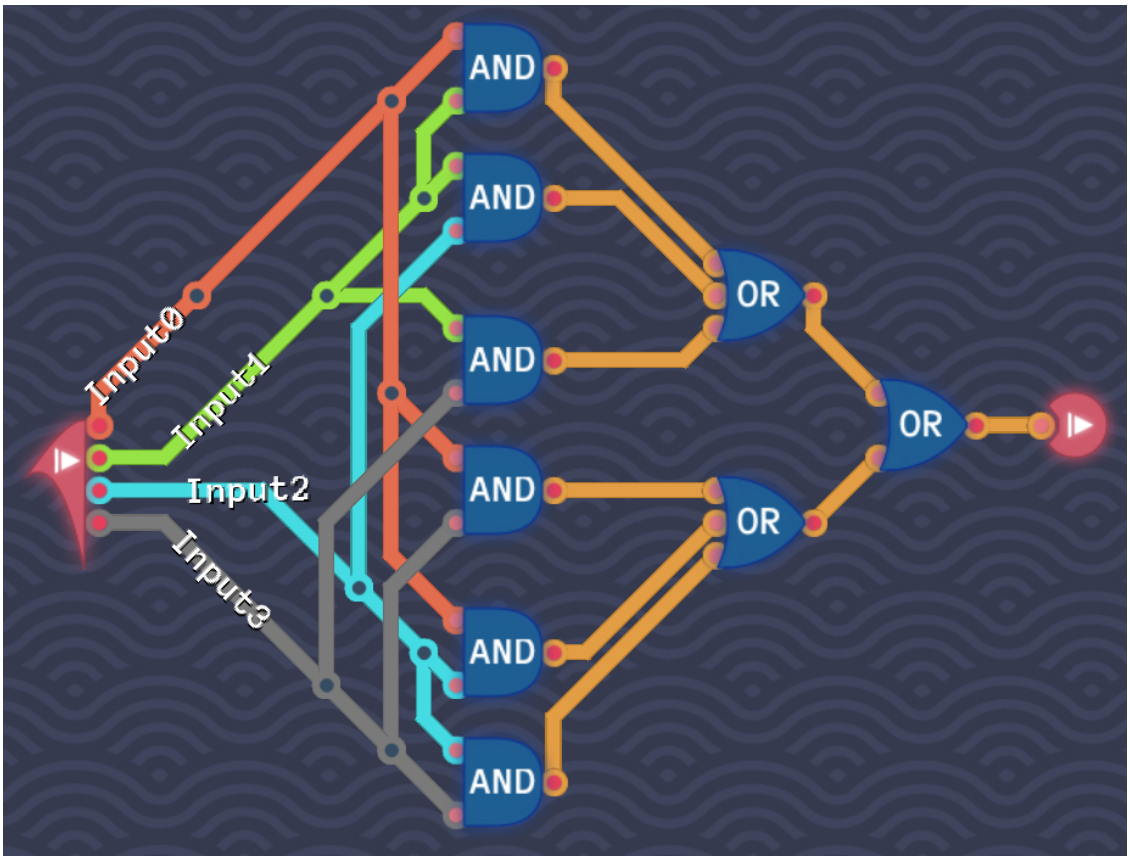


Figure 4: Decoder Puzzle Solution

in the top right, comprises a sequence of flip-flops. The addition process, involving one bit per high input bit, is deconstructed into half adders culminating in a single XOR at the end (Figure 6). This demonstrates to students that they do not need to use a full adder for every addition-like operation. Knowing and understanding the meaning and amount of inputs has a significant impact on how well a student can reduce the hardware cost.

Following the reading of the input by the shift register for one clock cycle, the shift register undergoes 32 clock cycles to sequentially feed into the accumulator. Once the series of additions concludes, the resultant values are directed to a straightforward two's complement comparator. The input bits in the bottom left undergo inversion and subsequent addition by one (Figure 7a). This process is completed by the implementation of a comparator through addition (Figure 7b). As only the most significant bit is pertinent in the comparison, the adder components are simplified to a basic c-out implementation.

This modular and efficient solution is conducive to scalability and can be presented to students after their individual submissions. The game additionally provides a gate score and critical path delay, serving as quantitative metrics that aid in pinpointing areas for enhancement and optimization in students' designs.

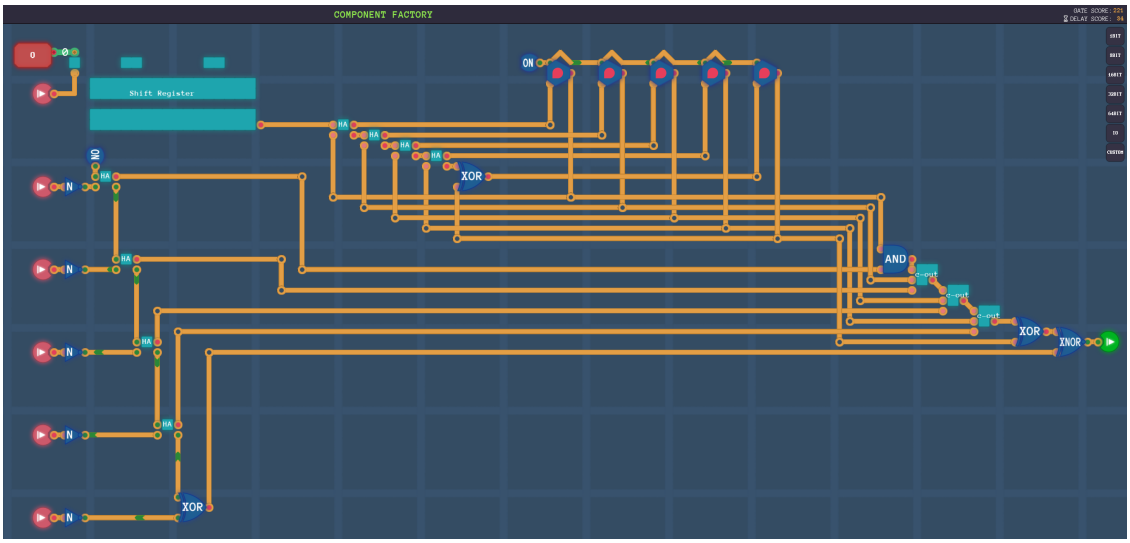


Figure 5: Our Scalable Solution

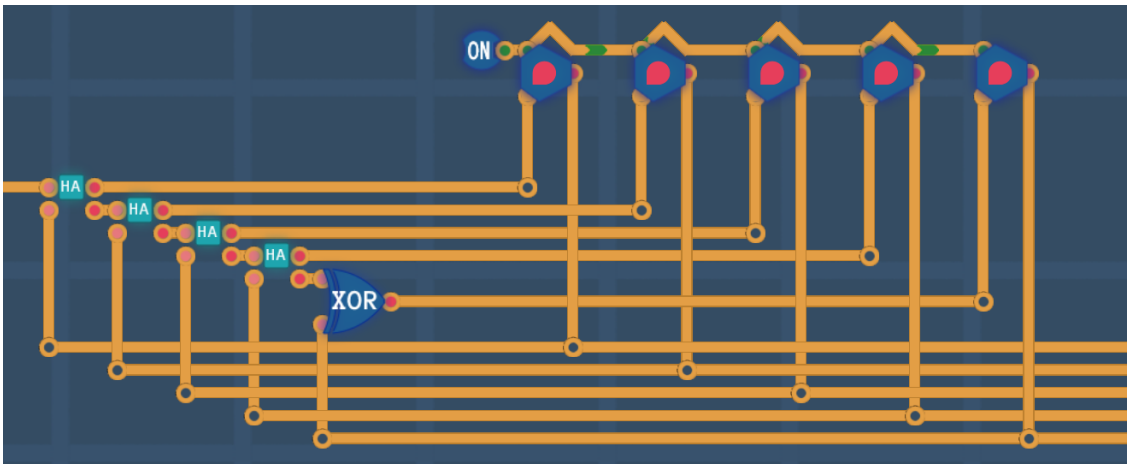
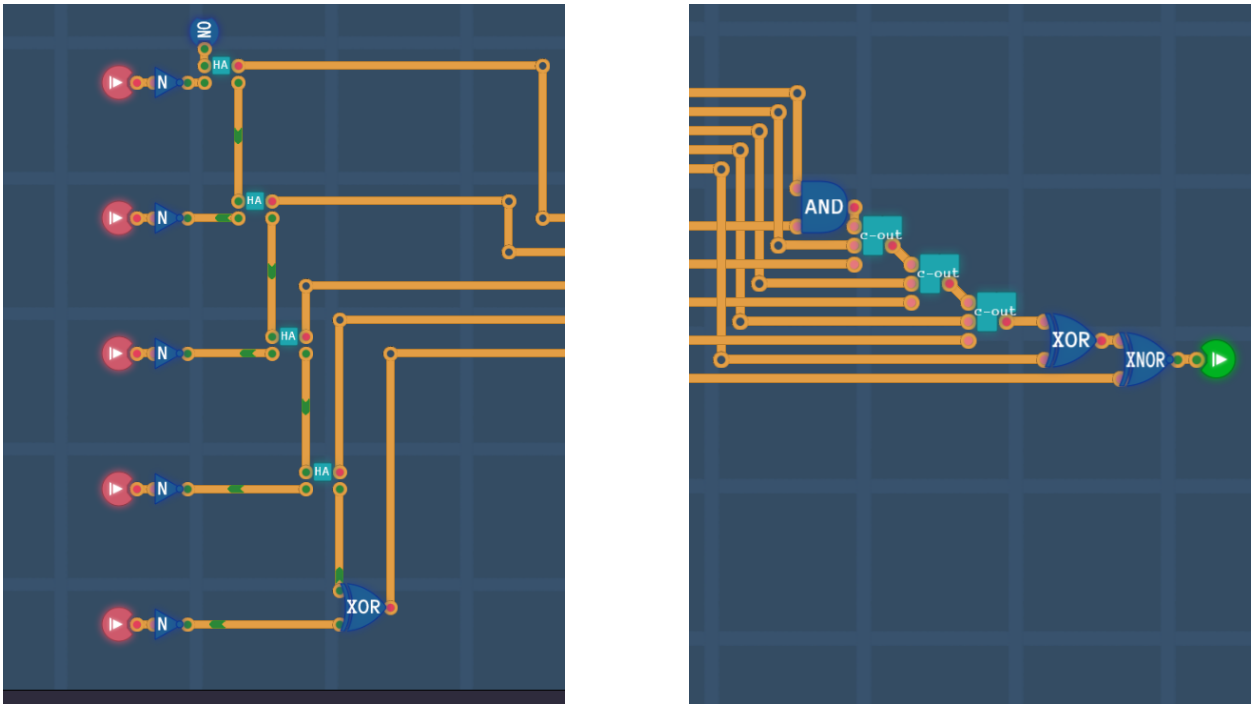


Figure 6: The Optimized Accumulator

Challenging Students Beyond Digital Logic

Upon the conclusion of the digital logic course, students are confronted with a one-year hiatus before the commencement of the computer architecture curriculum. This temporal gap often results in the gradual erosion of the material learned, leaving students disoriented when encountering analogous structures in computer architecture that presuppose a solid foundation. To mitigate this challenge and proactively cultivate students' comprehension of diverse processors, we propose an additional credit exercise to be undertaken before the initiation of the computer architecture course.

This exercise entails reaching the "Working Computer" stage, positioned prominently at the apex of the skill tree, and successfully implementing its solution. This endeavor serves a dual purpose: firstly, it enables students to sustain and deepen their hands-on experience with the fundamental principles of digital logic, fostering a continuous engagement with the material. Secondly, it



(a) Two's Complement Negation of 'n'

(b) The Optimized Comparator and Output

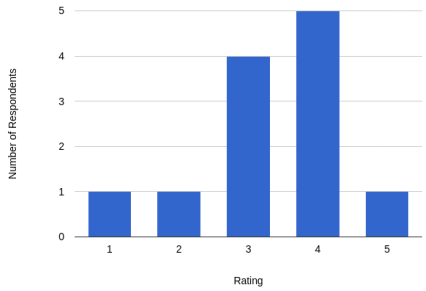
Figure 7: Full Comparison Process

strategically readies them for the implementation of a single-cycle processor, a focal point in the forthcoming computer architecture curriculum. Proficiency in this foundational concept seamlessly transitions into the exploration of other types of processor cores.

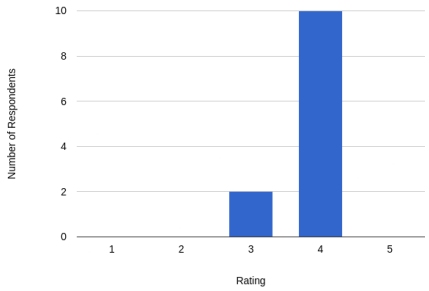
Students are incentivized to attain the "Working Computer" stage by the intrinsic value of practical experience, the prospect of earning extra credit at the onset of the computer architecture course, and the gratification of witnessing their self-designed 8-bit processor successfully execute machine code. This initiative not only reinforces and sustains digital logic competencies but also sets an advantageous trajectory for students as they embark on the more advanced intricacies of computer architecture.

Surveying Students' Perception of the Game

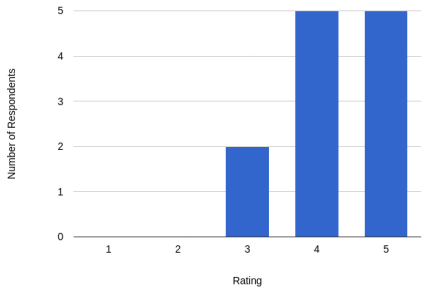
We conducted a survey among twelve of the Computer Architecture students following a demonstration of the game. The aim was to discover whether students looked favorably upon an implementation of the game in Digital Logic. The responses were gathered on a scale from 1 to 5, where 5 indicated a highly positive response and 1 signified a highly negative response.



(a) Likelihood to Purchase the Game



(b) Usefulness of Challenges in Exams or Labs



(c) Usefulness as Optional Supplemental Material

Figure 8: Survey Results

Discussion

The results of the survey were overwhelmingly positive, with one respondent indicating that they had purchased the game upon the conclusion of the demonstration. This sentiment is reflected in Figure 8a, where a surprising number of students appeared to be willing to pay the $20 dollar price tag for the game. Furthermore, students indicated that they would have found the game useful both as a source for material in the curriculum (Figure 8b) and as a supplemental tool to aid in their comprehension during Digital Logic (Figure 8c).

In the open response section of the survey, one student noted that caution should be taken in using challenges in the game directly in the graded course material as it would be relatively easy to find solutions online. Since our aim is to derive problems and discussion areas from the game instead of copying its puzzles directly into coursework, this would not be an issue, but it is important to take into consideration.

Conclusion

This paper delves into the integration of the Turing Complete game into digital logic and computer architecture education. Exploring challenges within the game, from decoders to 8-bit processors, we sought to bridge theory and practice. The proposed cross-curricular challenge, strategically positioned between digital logic and computer architecture, serves as a dynamic link, translating knowledge into hands-on application and essential preparation.

The game's sandbox and simulation features offer a space for students to experiment, adding a more tangible layer to traditional tools. Survey results indicate positive student views, with interest in purchasing and recognition of its value as supplemental material. However, caution arises in considering solution availability, urging careful implementation and a potential need for problem revision.

The Turing Complete game invites users of Vivado and ModelSim to explore simpler alternatives, with graphical interfaces enhancing the learning experience for novice students. Moving forward requires a balanced approach, as the game has its limitations, and students still need exposure to industry-standard tools. However, this game is positioned to reshape digital logic and computer architecture education, making it more engaging and enjoyable without sacrificing technical benefits. My professor has already begun implementing the ideas presented in this paper into our curriculum this year, driven by its popularity among my peers. Hopefully, this initiative will inspire other educators to consider integrating the game into their own courses. Not only could this provide additional data on its efficacy, but it would also offer other students the opportunity and experience I had.

Bibliography

[1] *Turing Complete*. Accessed: February 28, 2024. URL: https://turingcomplete.game/.

[2] *Steam*. Accessed: February 28, 2024. URL: https://store.steampowered.com/about/.

[3] Kingsley Ofosu-Ampong. "The Shift to Gamification in Education: A Review on Dominant Issues". In: *Journal of Educational Technology Systems* 49.1 (2020), pp. 113–137. DOI: 10.1177/0047239520917629. eprint: https://doi.org/10.1177/0047239520917629. URL: https://doi.org/10.1177/0047239520917629.

[4] Michael Sailer and Lisa Homner. "The Gamification of Learning: a Meta-analysis". In: *Educational Psychology Review* 32.1 (2020), pp. 77–112. ISSN: 1573-336X. DOI: 10.1007/s10648-019-09498-w. URL: https://doi.org/10.1007/s10648-019-09498-w.

[5] Rui Huang et al. "The impact of gamification in educational settings on student learning outcomes : a meta-analysis." In: *Educational Technology Research and Development* 68.4 (2020), pp. 1875–1901. ISSN: 10421629. URL: https://www.jstor.org/stable/48727510.