Capstone Proposal

Aaron Gordon

California State University Monterey Bay

Contents

Executive Summary

We propose a unique pedagogical approach to teaching the subject of computer architecture targeted at the layman and the beginning student: a rudimentary hardware simulator presented to end users as a puzzle game. Players will be introduced to basic circuit design, transistors and finally logic gates through a series of puzzles that ask them to build each component using only primitive parts (wires, batteries) and previously built components. Once a complete set of logic gates has been built this way players are tasked with the creation of the ALU of a processor - the "brain" at the heart of any modern computer - and then introduced to numerical systems of representing letters, words, images and more in numerical means the ALU can understand and perform work upon.

Although much of this work will seem trivial to students of computer science and computer engineering the beginning student and the layman will benefit greatly from a clearer mental model of how their home's power outlet is turned into an email message, a digital photo or anything else their computer creates. The presentation of material as a puzzle game, akin to Tetris, Bejeweled and other geometric pattern-recognition games, is crucial to making it accessible and engaging to a non-technical audience. To that end, hardware simulation will be controlled entirely via graphical representations of constituent parts that can be ergonomically dragged, dropped and wired to one another in a sort of digital breadboard. No hardware description languages or composite boolean functions will be presented.

When concluded this project should result in the creation of a standalone applications users can execute locally to play the game described above. Users will ideally finish the game with a deeper understanding of how modern electronics function on an intuitive level. It is our hope this understanding will instill in the layman an appreciation for the elegant and surprisingly simple systems that power our modern digital world.

Introduction

The advent of the "Internet of Things" is resulting in the integration of digital computers into more and more everyday products. Despite the growing ubiquity of such technology the physical mechanisms that undergird it remain a mysterious, even magical, black box to most users. Though the average computer user may have a firm understanding of how to operate their computer's software and may even grasp the basics of electrical current there still exists a gulf of knowledge that explains how the latter is used to generate the former.

This gulf is typically termed Computer Architecture, and herein we propose a unique pedagogical solution to bridging it in the public mind: a rudimentary hardware simulator presented to end users as a puzzle game. Players will be introduced to basic circuit design, transistors and finally logic gates through a series of puzzles that ask them to build each component using only primitive parts (wires, batteries) and previously built components. Once a complete set of logic gates has been built this way players are tasked with the creation of the ALU of a processor - the "brain" at the heart of any modern computer - and then introduced to systems of representing letters, words and images in numerical encodings the ALU can understand and perform work upon.

Much of the work detailed above will appear trivial to the computer science or computer engineering student – and rightly so. This material encompasses a holistic understanding of the physical properties at play in the construction and operation of a computer, and while such knowledge is considered elementary within the discipline it is sadly unknown without. Computers have unquestionably become a fundamental part of modern living in the developed world and continue to play a major role in the advancement of developing nations, as well. It's important that even those with a non-technical background can learn about the elegant and

powerful principles underlying the modern computer. It's this group of non-technical users – or laymen – that this project seeks to serve.

The way the human heart pumps blood through the circulatory system or the heliocentric nature of our solar system are processes understood by those outside the disciplines of medicine and astronomy, yet the manipulation of digital data via logic gates fundamental to modern computing remains practically a trade secret. Perhaps this reality is a result of the relative youth of the computer science discipline, or perhaps its because our tablets and our phones interest us far less than our bodies and our place in the universe. Whatever the reason, it is clear from an anecdotal observation of popular knowledge of the sciences that computer science is particularly poorly understood.

A wealth of textbooks, popular science books, do-it-yourself microcontroller kits, children's toys and even computer games have been produced throughout the years with the goal of educating the public about computer science. Each has its strengths and weaknesses, and each is limited by the nature of its medium.

Textbooks are excellent sources of knowledge, but typically require a wealth of prerequisite knowledge that makes them inaccessible to the layman. Popular science books are accessible in a way text books are not, but often lack the detail and hands-on experience of the classroom setting a textbook is typically used in. Microcontroller kits and board games provide an excellent hands-on experience yet tend to simplify or obfuscate important concepts for the purposes of ease-of-use; microcontroller kits can also require prohibitively expensive investment in physical components. Computer games can provide a limitless supply of virtual components to play with, but play is precisely what most prioritize at the expense of educational value.

Perhaps most conspicuously, presently available computer science education resources targeted at laymen are focused on a specific element of a computer's function, such as electrical circuits or writing microcontroller code. It would seem only resources targeted at the technician seek to describe the computer's physical operation in its totality. Where the proposed project will differentiate itself is in the combination of the accessibility of toys, games and popular science books with the holistic view of academic texts.

Project Goals

- Balance technical accuracy with understandability
    - Require no prerequisite knowledge from users
    - Require no access to out-of-game resources, such as guides or manuals
- Provide a holistic view of a digital computer's physical operation
- Prioritize education over play
    - Do not distract with peripheral game elements
- Minimize upfront investment
    - Require no physical parts
    - Distribute as freely as possible

Project Objectives

1. Implementation of a rudimentary hardware simulator as a C# library
    1.1. Support for basic circuit design using transistors
    1.2. Support for logic gate design
    1.3. Support for composite logic gates (i.e. gates built from simpler gates)

2. Development of an ergonomic, graphical user interface for controlling the hardware simulator using Unity 3D
    2.1. Support for “drag and drop” placement of components
    2.2. Support for visually wiring output from one component to the input of another
    2.3. Support for visual feedback of the voltage in a wire (i.e. high/low, on/off)
    2.4. Support for the display of human-friendly conversions of binary values (i.e. integers, characters, pixel colors)
3. Design of a series of intuitive puzzles that lead players through the creation of a simple computer using the hardware simulator and user interface
    3.1. Puzzles focused on transistor behavior and the creation of basic logic gates
    3.2. Puzzles focused on basic logic gate behavior and the creation of advanced logic gates
    3.3. Puzzles focused on representing characters, words and images using bitwise encoding
    3.4. Puzzles focused on the creation of a simple ALU using previously built gates
    3.5. Puzzles focused on manipulating characters, words and images using the ALU

Literature Review

Neither outreach to the public nor the application of the computer itself are novel approaches to computer science education. We have identified a collection of excellent, presently available and proposed resources for computer architecture education that share one or more of the stated goals of this project. These resources are discussed below in association with the strengths and limitations implicit in their respective mediums.

Perhaps the most obvious resource one turns to in the pursuit of knowledge is the book. An important distinction should be made between textbooks written for academic purposes and what we have labeled popular science books intended for public, non-academic consumption.

One of the more popular textbooks on computer architecture is Patterson and Hennessy's Computer Organization and Design (Patterson 2018). This is a thorough introduction to the hardware-software interface that touches not only on the physical properties of the microchips and microprocessors that power digital computing but on their manufacture and fabrication, as well. Alas, Patterson and Hennessy's text, written as part of the duos teaching at Berkley, makes a great number of assumptions about the knowledge of its readers – knowledge which cannot be assumed in the public.

A less presuming text is Nisan and Schocken's The Elements of Computing Systems (Nisan, 2005). This text assumes little about its readers preexisting knowledge of Boolean logic and arithmetic but emphasizes hands-on labs that require programming in hardware definition languages. The lab work is integral to the book's pedagogical approach, but would no doubt prove too intimidating to the typical layman who's never written a line of code.

A plethora of popular science books on computer architecture exist. Popular and typically well-reviewed on the Amazon.com bookstore is Code: The Hidden Language of Computer Hardware and Software by Petzold (Petzold, 2015). This text is targeted specifically at the public and draws on well-known historic analogies like Morse code to explain its concepts. However, Petzold's work is limited by its medium: as powerful as the written word can be it often fails to communicate what a more tactile experience is able to.

To that end several mini computers and microcontrollers intended for educational purposes have been developed. The two most popular representatives of each category are the

Raspberry Pi and the Arduino, respectively. Both products offer users the opportunity for hands-on experience with digital computing at a much lower level than the typical consumer product. Yet this experience is not quite low level *enough* – both products effectively function as black boxes with programmable inputs and outputs that, while powerful, fail to expose the underlying physics at play. In addition, by virtue of being physical products there is a steep, often prohibitive cost to experimenting too brazenly with either, as any broken parts need replacing.

Attempting to reduce the cost of experimenting with physical hardware several developers have begun creating virtual breadboards. Applications such as Autodesk Circuits allow users to drag and drop electrical components into a virtual workspace and then simulate their behavior. These tools provide a low cost (often free) means of learning to work with microcontrollers like the Arduino and do an excellent job of digitally replicating the tactile experience of wiring input and output pins. The same limitations of microcontroller boards apply to these programs as well, though: namely obfuscation of the hardware-software interface for ease-of-use.

Countless toys and games have been produced that attempt to further distill the hands-on experience of boards like the Raspberry Pi and Arduino into something simultaneously fun and educational. Popular with young children are "circuit building blocks" that allow the fabrication of various circuit designs by snapping large metal connectors of predetermined length to one-another. Particularly noteworthy is a board game designed by Ben Heck that focuses on logic gate design, though unfortunately the future production of said game remains uncertain (E. 2014).

Unsurprisingly, computer game programmers have at times also taken inspiration from the discipline of computer architecture. The popular building game Minecraft by developer Mojang includes a special kind of brick called "redstone" that behaves like a semiconductor (Elliott, 2017). Puzzle games such as Suborbital Games' Circuit Scramble present players with prebuilt logic gates that must have their inputs properly arranged in as few moves as possible to achieve a desired output (Staalduinen, 2016). Perhaps most like the proposed work of this project are the games of developer Zachtronics. In Shenzhen I/O players assume the role of a computer engineer tasked with designing and programming integrated circuits.

All of these games, though – both cardboard and digital – are preoccupied with play. Minecraft's redstone, for example, is just one of many kinds of bricks players can place in their world, making its usefulness as an educational tool limited to a carefully curated play session. Zachtronics' games emphasize the experience of working as a programmer in the early days of the industry; indeed, many of the games are intentionally obtuse precisely to achieve an atmosphere of confusion. Shenzhen I/O in particular includes a 30-page manual of datasheets and assembly language specifications players are encouraged to print out for the authentic 1980s experience. The common thread, then, is that these games require a teacher to turn them into educational experiences.

End User

The software artifact resulting from this project will be targeted at two primary types of user, 1) the non-technical layman interested in understanding his or her computer's operation on a physical level and 2) the beginning computer science student interested in understanding the hardware-software interface (i.e. how does electricity become executable code). Both types of

individual are assumed to be familiar with basic computer operation but unexposed to technical concepts such as transistor behavior, circuit design, boolean logic, boolean arithmetic, hardware description languages, machine languages, etc. A third category of user - recreational players of puzzle games - may also show interest in the final project for its value as a series of brain teasers, however this project is not being pursued with them explicitly in mind.

If successfully executed this project will generate value for stakeholders through the provision of an accessible, holistic education on the physical operation of a digital computer. As discussed in the above introduction the ubiquity of digital computers is unbalanced with popular understanding of their operation. Though many of the materials discussed in the literature review above already provide a similar education, none appear to offer both a balance of technical accuracy with understandability and a complete picture of how electric current is harnessed by logic gates to construct a processor capable of performing the operations we've come to expect from our laptops, tablets, phone, televisions and more.

Methodology

The successful execution of this project hinges on thorough understanding the material, careful consideration of a pedagogical approach and technical ability to execute on the development of the specific software artifacts.

The material discussed in the above literature review will provide an excellent source for reviewing the material being covered in this project. Reading or rereading literature on the topic will shore up any misunderstandings or gaps in knowledge. More importantly, taking time to play with virtual breadboard software and computer games that draw their inspiration from computer architecture will provide an opportunity to understand how other developers have

tackled these topics, both from a technical implementation standpoint and from the important perspective of making the subject matter fun and engaging for players. This will also provide the opportunity to identify where existing software fails in educating its users, either by failing to cover specific elements of the proposed material or by overemphasizing game play or aesthetic over pedagogy.

In considering pedagogy perhaps the most careful thought must be given to the order in which topics are introduced. A certain order is naturally inherent in the material as we build a basic ALU from the ground up: circuits must precede transistors, which in turn must precede logic gates and so on. However, what order should individual types of gates be introduced? Should a working version of a gate be given to users to experiment with before they're asked to construct one themselves? The answers to these sorts of questions will ultimately inform the design of the puzzles presented to players in the final artifact and, in turn, its effectiveness as an educational tool. Finding answers to these questions, then, is a crucial step that should be considered and reconsidered starting early in the project.

Of course, the best pedagogy is useless to this project if it's never implemented. Technical implementation will begin with research into the software architecture of hardware simulators, a deep and well researched classification of software. Fortunately, the hardware simulator required for this product is exceptionally simple and so adequate literature should be easy to come by. Parallel to implementing a hardware simulator a graphical frontend to accommodate intuitive user interaction with the simulator must be developed.

This frontend will be developed using Unity 3D, which was selected in part for its large, existing libraries designed to handle user interactions like mouse clicks and view scrolling. Unity 3D exposes its libraries for development in the C# language, which makes the

development of a hardware simulator as its own C# library a natural fit. As the hardware simulator library has new functionality implemented within it, a simple integration layer can take advantage of newly exposed methods in combination with Unity 3D's frontend libraries to quickly prototype and test user interaction with the hardware simulator. Finally, Unity 3D ships with a powerful level editing tool that will assist in the creation of puzzles using the integrated hardware simulator / frontend components.

Ethical Considerations

The primary ethical consideration with regards to the proposed project is the respect of the intellectual property of other programmers and educators. As evidenced by the extensive literature review performed above there already exists a great wealth of educational material on the topic of computer architecture. Indeed, the very essence of this project is not the conveyance of new knowledge but of well-understood knowledge to a previously unreached group. As such there will be inevitable similarities between the software artifact resulting from this project's work and existing texts and computer games. However, care must be taken to minimize these similarities by the thorough application of a unique pedagogical approach to the topic, not only for the benefit of the user but out of respect to the holders of the intellectual property rights to the aforementioned material that has inspired this project.

It will also be important to consider a broad group of individuals when developing the pedagogy of this project. If digital distribution of the final software artifact is made possible then potential users may come from diverse socio-economic backgrounds with diverse, pre-existing knowledge of the topic. What may seem like common knowledge to an individual from

one group can be a stunning revelation to that from another. It's important that presumptions of the user's knowledge are kept to a minimum.

It would be beneficial instead to attempt to *predict gaps* in a hypothetical user's knowledge. The abstract idea that electric current "flows" like water through a pipe is not learned through everyday observation of electrons (were such possible) but in a grade-school classroom. Yet there are many who have not enjoyed the privilege of access to high quality grade-schools. The flow of current and other related ideas needs to be at the very least touched upon for these users.

No doubt other life experiences that help paint the mental models computer architects rely upon to learn the topics addressed by this project are often taken for granted. Careful consideration must be given to what these experiences might be and what lessons they might impart, and concessions for those who may not have had such experiences should be made where possible. Drawing early feedback from as broad a group of playtests as possible may prove invaluable in addressing this consideration.

A final consideration, though minor, is distribution. If the software product resulting at the end of this project does indeed show promise for educational use, then it can ideally be made available for download free of charge. This arrangement may need to be discussed with any employers of project team members at the project's conclusion.

Legal Considerations

Two pieces of third-party software will be leveraged in the development of the proposed project: Unity 3D and Autodesk Maya.

Unity 3D is freely available for commercial and non-commercial use with some limitations (notably the automatic inclusion of a Unity 3D splash screen at the startup of applications built with the platform). These limitations may be removed from the product via the purchase of a professional license. However, since these limitations pose no obstruction to the completion of the project described herein the free version of the software will serve adequately.

An educational edition of Autodesk Maya is available to students for non-commercial use with no notable limitations. The proposed project meets the criteria of student work necessary for the use of the educational version, and the intended free distribution of the resulting software artifact insures the project remains non-commercial in nature.

Project Scope

The following schedule outlines planned work towards the completion of the proposed project on a week-by-week basis. Important milestones are denoted in the rightmost column. Work has been organized in vertical slices where possible to allow the identification and resolution of integration issues as early in the project as possible. Dates have been estimated as carefully as possible but may be subject to change.

Week	Objectives	Tasks	Milestone
1	1.2, 2.2, 2.3	Implement built-in basic logic gates (NOT, AND, OR, XOR) in hardware simulator. Implement rendering of components, wires, voltages in GUI application.	
2	1.3, 2.1, 3.2	Implement composite logic gates in hardware simulator. Implement drag and drop behavior in GUI application. Design advanced logic gate construction puzzles.	First Working Prototype
3	1.1, 2.1	Implement circuits in hardware simulator. Implement support for new components in GUI application.	
4	1.1, 3.1	Implement transistors in hardware simulator. Design basic logic gate construction puzzles.	
5	1.2, 2.4, 3.3	Implement built-in advanced logic gates (MULTIPLEXOR, ADDER) in hardware simulator. Implement human-readable displays in GUI application. Design bitwise encoding puzzles.	GUI Application Complete
6	1.2, 3.4	Implement built-in ALU in hardware simulator. Design ALU construction puzzles. Begin test-group surveys.	Hardware Simulator Complete
7	3.5	Design bit manipulation puzzles. Fix bugs. Adjust puzzles from survey feedback.	Puzzles Complete
8		Fix bugs. Adjust puzzles from survey feedback. Distribution, if applicable.	Project Complete

The proposed project requires rendering a virtual circuit board to the end user's display. This will be accomplished using Unity 3D, a freely available application designed for the creation of computer games and similar software. Rendering components such as power sources, transistors and integrated circuits as part of the virtual board will require vertex data that represents their specific three-dimensional shape. This vertex data will be generated using an educational edition of Autodesk Maya, a modeling application used in the creation of films and computer games. Please refer to the section titled Legal Considerations for additional details on the usage of both Unity 3D and Autodesk Maya.

Given the complex, multilayered nature of the proposed project there exist several risks to its punctual completion. The primary risk is the unknown time requirements for designing and implementing the various proposed puzzles. The exact number and complexity of puzzles has

been left open-ended for this reason. This risk is being additionally mitigated by spreading puzzle design and implementation tasks throughout the entirety of the project's schedule, as can be observed in the above table.

A similar risk is the complexity of implementing both a hardware simulator and an interactive, game-like user interface. This risk is being mitigated via the use of a known, existing technology designed to simplify the latter – Unity 3D – and by scheduling completion of both the hardware simulator and the GUI application earlier in the project's schedule (weeks 6 and 5, respectively) to allow work on these features to "spill over" into the back quarter of the project, if needed.

Finally, the usability of the completed software artifact is a risk in-and-of itself. This risk will be mitigated through continued software testing with a pre-arranged survey group during the last three weeks of development. Feedback from this group will be integrated into each weekly iteration of the project to improve its usability as both a hardware simulator and an educational tool.

Final Deliverables

There are three primary deliverables for the proposed project: the hardware simulator software library, the playable puzzle game (dependent upon said library) and a single website for describing and distributing both together.

The hardware simulator software library will take the final form of a Microsoft Dynamic Link Library (.DLL) file. The library will be implemented in C# and intended for use with other C# and .NET / Mono applications. The "virtual board" and its constituent components will be exposed to consumers of the library via public interfaces, most probably in the form of C# class

definitions. If possible, this library will be freely open-sourced via GitHub or a similar code sharing platform both during and at the conclusion of its development.

The playable puzzle game will be a standalone executable generated by Unity 3D and dependent upon the hardware simulator software library. Unity 3D applications are compiled against the Mono framework, which will allow development of the puzzle game in the C# language to expedite consumption of the hardware simulator software library. Like the software library, the code used to build the final executable will be freely open-sourced if possible.

The final deliverable is a website used for distributing the playable game. Both the game and software library code will be made available to the public via open sourcing, however this requires end users to have the requisite tools and knowledge to compile the source code into a final product. Given the target audience of the project is the non-technical layman this is an unacceptable distribution method. Instead a website will be created that provides download links directly to the standalone executable with a description of the project and its intended use.

Evaluation

Evaluating the usability of the software artifact resulting from the proposed project will require two dimensions of measurement. First, the software must be evaluated for usability as a hardware simulator. Second, the software must be evaluated for its ability to educate its end users in the subject of computer architecture. Both of these axes will be measured by collecting feedback from a preassembled group of testers.

Evaluation of the final application for usability as a hardware simulator will require feedback from professional engineers and programmers familiar with similar applications. This feedback will be collected via a survey given to members from the test group that have

appropriate experience in these areas, or similar experience with software of equal complexity. Participants will be asked to identify any obstacles they encountered in performing certain tasks in the application, either due to a lack of clarity in the user interface or as a result of unexpected behavior in the application.

Evaluation of the project as an educational tool will require feedback from non-technical laymen who had little or no prior experience with the subject of computer architecture. This feedback will also be collected via survey from members of the test group that match these qualifications. Participants will be asked how many puzzles they were able to complete, what puzzles – if any – proved particularly difficult to complete and how they would self-rate their understanding of the topics covered by the application.

Samples of both technical and non-technical surveys are available in Appendix A and B, respectively. These samples will change as specific GUI implementations and puzzle designs are clarified over the course of the project's development.

Team Members

This is an individual project proposed by Aaron Gordon, a student at California State University Monterey Bay. Mr. Gordon will be responsible for the entirety of the project's design, documentation, implementation and distribution. Additional assistance for testing the software artifact resulting from the project will be provided by a survey group assembled by Mr. Gordon explicitly for such purposes.

REFERENCES

About SHENZHEN I/O. (n.d.). Retrieved from http://www.zachtronics.com/shenzhen-io/

Arduino - Home. (n.d.). Retrieved from https://www.arduino.cc/E. (2017, August 04).

Ben Heck's Logic Gate board game: The finale. Retrieved from https://www.engadget.com/2017/08/13/ben-heck-s-logic-gate-board-game-the-finale/

Bring ideas to life with free online Arduino simulator and PCB apps | Autodesk Circuits. (n.d.). Retrieved from https://circuits.io/Elliott, R. (2017, December 1).

Redstone Circuits. Retrieved from https://education.minecraft.net/lessons/redstone-circuits/

Nisan, N. (2005). The elements of computing systems: Building a modern computer from first principles. London: The MIT Press.

Patterson, D. A., & Hennessy, J. L. (2018). Computer organization and design The hardware/software interface. Cambridge, MA: Morgan Kaufmann.

Petzold, C. (2015). Code: The hidden language of computer hardware and software. Redmond, WA: Microsoft Press.

Staalduinen, R. V. (2016, April 8). Circuit Scramble. Retrieved from http://suborbitalgames.com/?p=142

Teach, Learn, and Make with Raspberry Pi. (n.d.). Retrieved from https://www.raspberrypi.org/

APPENDIX A: USABILITY SURVEY (TECHNICAL)

1. Please rate your prior familiarity with each of the following on a scale from 1 to 5, 1 being no experience and 5 being a professional level of experience.

   Electrical Engineering: ___

   Circuit Design: ___

   Hardware Simulators: ___

   Software Engineering: ___

2. Please identify the most complex software applications you're able to use confidently, such as Computer Aided Engineering software, 3D modeling software, simulation software, etc.

   _____

   _____

   _____

3. Please note any elements of the user interface (buttons, panels, etc.) you found unclear.

   _____

   _____

4. Please note any function in the application the behaved differently than expected. Please include the expected as well as the actual behavior.

   _____

   _____

APPENDIX B: USABILITY SURVEY (NON-TECHNICAL)

1. Please identify your prior experience in computer architecture, if any.

   _____

2. Please list your favorite puzzle games, if any. Include computer and video games, tablet or mobile games and board games.

   _____

3. Please enumerate the highest puzzle completed: _____

4. Please note those puzzles that proved the most difficult to complete.

   _____

   _____

5. Please rate your confidence in your understanding of the following topics on a scale from 1 to 5, 1 being no confidence and 5 being absolute confidence.

   Electrical Current and Circuit Design: ___

   NOT, AND, OR and XOR Gates: ___

   Multiplexors and Demultiplexors: ___

   Bitwise Encoding: ___

   Boolean Arithmetic and Adder Chips: ___

   Arithmetic Logic Units: ___

   Bitwise Manipulation: ___