PROGRAMMABLE ANALOG CO-COMPUTING

Harris Coughlan McRae

Bachelor of Engineering (Hon.)
Software Engineering
Bachelor of Science
Astrophysics

School of Engineering
Macquarie University

June 2, 2024

Ansgar Fehnker, Carl Svensson, Damian Jurd

ACKNOWLEDGMENTS

Thank you to Ansgar Fehnker, Carl Svensson and Damian Jurd for agreeing to join my thesis as supervisors. It would not be possible without them.

STATEMENT OF CANDIDATE

I, Harris Coughlan McRae, declare that this report, submitted as part of the requirement for the award of Bachelor of Engineering in the School of Engineering, Macquarie University, is entirely my own work unless otherwise referenced or acknowledged. This document has not been submitted for qualification or assessment an any academic institution.

Student's Name: Harris C. McRae

Student's Signature: Harris C. McRae (Electronic)

Date: 02-06-2024

ABSTRACT

Digital computing hardware is arriving at a physical limit with transistors approaching sizes countable in atoms. This signals the need for change in computing technologies to keep up with the growing demands of software. One such option, considered primarily for application of neural networks, is integrated electrical-analog cells, either in-memory or as a peripheral. This paper aims to examine how analog computer components may be utilized in software tools for specific fields by identifying and implementing such processes on a simulated model of an analog co-computer.

Contents

Acknowledgments **iii**

Abstract **vii**

Table of Contents **ix**

List of Figures **xi**

1 Introduction **1**
1.1 Research goal 2

2 Background and Related Work **3**
2.1 Analog computing as a modelling tool 3
2.1.1 Operation 3
2.1.2 Anabrid Analog Computers 4
2.1.3 Analog compared to Digital 4
2.1.4 Components 6
2.2 Hardware acceleration 9
2.3 Modern Analog Computing 10
2.3.1 Components 10
2.3.2 Findings 10

3 Methodology **13**
3.1 Defining an abstract model 14
3.2 Implementing a simulation of the model 14
3.3 Identifying & implementing software processes in the simulated model . . . 15
3.4 Analysis 15
3.5 Summary 15

4 Model verification **17**

5 Results **19**
5.1 Analysis 19

6 Discussion & Future Work **21**

7 Conclusion **23**

8 Abbreviations **25**

Bibliography **25**

List of Figures

2.1 *An abstract representation of a summer on two inputs. Note the third input, known as a summing junction, used in most op-amp summers to provide negative feedback necessary. The summing junction is tied to the negative rail of the output.* 6
2.2 *An abstract representation of an integrator with three inputs. A fourth input, the initial condition* $e(0)$*, is also shown.* 7

Chapter 1

Introduction

Ever since the inception of the digital computer, there has been a continuous forward drive to create more powerful digital computers. This is articulated by Gordon Moore with his famed Moore's Law [11], predicted back in 1965 - that the number of transistors on an integrated computer chip would double approximately every 2 years, which has largely held true since. However, over the last decade signs have started to show that this trend is coming to a close, with the size of transistors reaching sizes where quantum phenomena such as electron tunneling begin to degrade their efficacy [11].

To maintain an increase in performance, hardware acceleration - computer hardware designed to handle specific tasks at a higher efficiency & performance than a general-purpose processing unit - has become increasingly prevalent in modern computer systems [6, 7]. This includes dedicated hardware such as GPUs and FPGAs, both allowing either high bandwidth or high efficiency in computation through utilizing configurable hardware components unlike a CPU, and mixed-signal circuitry for modems & other telecommunications processes that capitalize on the efficiency of operating directly on analog signals.

In recent years, GPUs and similar technologies have seen great use for AI and neural-network applications thanks to the parallel computing architecture featured. With the widespread usage of complex neural networks for applications such as LLMs or autonomous functions, concerns over the scalability of GPU hardware for neural network applications have arisen - primarily concerning energy efficiency and performance.

To address these problems, new forms of hardware are being researched and developed in the form of electronic analog computers. These systems, often integrated directly into a computer's memory or CPU, operate continuously, as opposed to a digital CPU performing discrete steps. This mathematical method of computing demonstrates promise for the future of a variety of neural network & similar processes in terms of both energy efficiency and performance moving forward. However, a large amount of research efforts have been put towards high-performance CNN-capable devices, while computer performance requirements as a whole continue to grow; despite analog machines being historically utilized to perform functions which have been since been adopted by digital

computers.

1.1 Research goal

The main goal of this paper is to identify some software processes which are applicable to the strengths of an analog computer. A form of a programmable digital-analog hybrid computing device will be proposed; this device will be implemented in a simulated library to allow it to be demonstrated and incorporated into example programs. The results of this simulation process will be analysed to determine the efficiency benefits of using the system, and the scale at which this system can be used will be discussed.

To accomplish this, an analysis will be performed of the components of an analog computer, with reference to modern applications of integrated analog computers. From this, an abstract model of a configurable analog-digital hybrid computer will be produced. This model will be verified by comparison to real-world analog computers to ensure that it demonstrates the same capabilities and behaviour of the real-world example.

At this time, the specific strengths of an analog machine will be analysed so that a measure of 'analog fitness' can be developed to assist in deciding what processes could be implemented upon this system, as well as how to do so.

A library package will be developed that implements the simulated system, allowing implementation into software code for demonstration processes. The performance benefits may be predicted at this point, as well as ease of development.

Chapter 2

Background and Related Work

2.1 Analog computing as a modelling tool

Analog machines (or computers) have been historically hard to define. Largely, an analog machine is a model of some problem, with the computation element being a physical system that acts as an analogy for a modeled system; for example, the Phillips machine, a hydraulic (water-based) system designed to model the British economy of the late 1940s and early 1950s. The flow of water itself was analogous to the flow of money in a functional economy with various tanks designated as banks, funds, spending, national input & output, and so forth [3]. Other applications are those of early NASA flight simulators, such as the X-15 flight simulator from 1963, which utilized a simple analog computer system to replicate the control and actions of the X-15 aircraft. This system was updated to include a digital computer to produce certain functions in 1965 [1].

Most 'traditional' electronic analog computers are in one of two forms, which will be referred to as 'patch-panel' analog computers or 'integrated' analog computers. Patch-panel analog computers have multiple analog computer components spread across a panel, which can then be connected to each other using cables. As a result, these types of computers are easily re-programmable, although have to be done manually and with an understanding of how the components of an analog computer interact with each other.

Meanwhile, integrated analog computers implement the components of an analog computer into an immutable circuit. They can be considered analogous to digital logic implemented in an integrated circuit compared to a breadboard prototype — these variants generally bear better performance, efficiency, and higher accuracy, although accuracy is not intrinsic to analog computing.

2.1.1 Operation

As mentioned above, analog computers largely fall under two categories, which will be referred to as *patch-panel* analog computers and *integrated* analog computers.

Patch-panel analog computers utilize the titular patch-panel as a medium through

which the components of an analog computer can be connected to form a circuit - connections are made by connecting leads from component to component, allowing the signals travel. These models allow ease of modification, though do make up a larger volume, disallowing them from conventional or mobile use. Patch-panel analog computer boards can be, in some models, pre-patched and swapped out, allowing operators to 'save' configurations and cutting out the time needed to re-program the computer.

Integrated-circuit analog computers are static analog computers that are designed onto an integrated circuit board or other similar design for use in integrated systems or mobile devices [4]. These designs are currently less modifiable, but some approaches do allow limited programmability [8].

2.1.2 Anabrid Analog Computers

Anabrid, founded by Dr. Bernd Ulmann, is one of the leading companies in analog computer development for commercial purposes, being behind two products; That Analog Thing and the Analog Paradigm Model-1. These two analog computers fall under the earlier mentioned 'patch-panel' style analog computer, allowing user modification and programming at the cost of space and some performance detriments.

That Analog Thing (THAT) [2] is an open-hardware patch-panel analog computer the size of a book, designed primarily for education and research projects. It is unable on its own to compete with the scale of any larger formal analog computer, but thanks to the nature of a patch-panel, multiple THATs can be strung together to produce larger programs. The Analog Paradigm Model-1 is a similar system, though at a slightly larger scale with a modular nature. Small analog module cards can be inserted and connected into the main body of the computer to expand functionality.

2.1.3 Analog compared to Digital

Analog computers operate fundamentally differently to digital computers. Digital computers operate on algorithms with discrete steps in the form of program code. This discrete nature extends to the data a digital computer operates on, which, in binary form, has limitations on the accuracy to which something can be represented. Meanwhile, analog computers don't feature discrete algorithms or program memory, and instead operates on creating a mathematically-focused analogous representation of a system, functioning similarly to the physical model it represents.

As a result, most measures of digital computer performance (clock speed in hZ, core count, threads, et cetera) do not apply to an analog computer per se as the two operate under different paradigms. Most analog computers are numerically compared to digital computers in terms of two statistics: energy efficiency (watts, W), or floating point operations per-second (FLOPs). Analog computers tend to outperform digital counterparts in these areas:

- Analog computing demands much less power than their digital counterparts (FPGA, GPU, CPU) while performing the same amount of work [4, 9, 10].
- Analog computing maintains a greater ratio of FLOP to joules (1 joule is equivalent to 1 watt per second; FLOP/J is equivalent to FLOPs/W) on equivalent systems - with Anabrid's *Analog Paradigm Model-1* displaying a ratio of $7.5 \times 10^{8\pm1}$ FLOP/J compared to a single-core desktop CPU at 1×10^8 FLOP/J [9]. Similar results have been found with modern integrated analog computers [10].

Other advantages of Analog Computing

Analog computers are applicable for any deeply-integrated process that operates on reading signals from an external peripheral - for example, smart devices and sensors or monitors. With digital devices, the incoming signal often requires parsing through an ADC to be operated on by a digital computer, a process that takes a distinct number of steps. An analog computer can operate upon the input signal and produce an output signal without any conversion in potentially faster time with a greater energy efficiency.

Disadvantages; Accuracy

Some implementations of analog computing in the modern age have been coined approximate computing due to a considerable margin of error in outputs. This is an artifact of analog computing as a whole, not present in digital computing due to the discrete nature of a digital computer (a digital signal lies in two ranges, with a 'high' range and a 'low' range - compared to an analog signal, in which the signal is the data itself, rather than a means of transmission).

These error bars come from noise inherent to the medium the analog system exists in, as well as, historically, errors in measurements of the analog machine itself. Digital computers are resistant to these errors, as a marginal percentage of error in a digital signal transmission is still readable as the original data to be re-transmitted or operated upon; however, as analog computers do not have discrete ranges (see above), the marginal error compounds throughout the components of the analog computer. This error renders analog computing less optimal for tasks that require extremely high precision.

Disadvantages; Standardization & repeatability

In a similar vein to accuracy, analog computers are also subject to problems regarding the ability to reproduce results. Given that slight changes in equipment or environment can cause errors that compound onwards, performing the same computation with two analog computers, or at two different times, can produce distinct results. With computers being found across the globe and often communicating with each other, this has the chance to create larger and larger errors which must be considered in programming.

2.1.4 Components

Machine Units

Generally speaking, an analog machine operates on a range of +M to -M, known as a machine unit. Machine units are arbitrary and interpreted as whatever is most convenient for the machine itself, often M=1.
In electrical analog machines, voltages are naturally the medium for machine units, with the voltage of a given cell being $\pm K$V, while still being treated as a value +M to -M. The range M can be scaled to problem at hand, and should be considered so that machine units do not exceed $\pm M$ while making the greatest use of the range. Generally, overloaded units will cause a halt of an analog machine.
In most machines, the hardware unit ($\pm K$V) and the machine unit ($\pm M$) should be treated as separate factors; the hardware unit is a function of the operational voltage and is the medium through through which the arbitrary machine unit is represented; hence, the machine unit is often considered ± 1. [12]

Summer

As the most basic unit of an analog machine, a Summer acts to simply represent the *negative* sum of two (or more) input values. Each input to a summer unit can have a fixed weight as a multiplicative factor - typically 1 or 10.

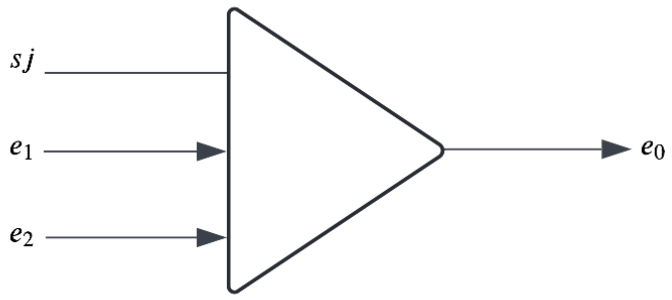

Figure 2.1: *An abstract representation of a summer on two inputs. Note the third input, known as a summing junction, used in most op-amp summers to provide negative feedback necessary. The summing junction is tied to the negative rail of the output.*

A summer can be described mathematically with the following form:

$$e_0 = -\sum_{i=1}^{n} a_i e_i$$

In this form, e_i is the input i, a_i the weight at that input, and e_0 being the *inverse* sum of all n inputs.

Integrators

An integrator performs an integration of multiple elements with respect to time, yielding the negative result similar to a summer. With an integrator, all inputs count as a rate of change; on an integrator with an input value of 1, the integrator will approach 1 over the span of 1 time unit. The final value held by the integrator is with respect to the history of the inputs since the integration began — hence, an integrator can be treated similarly to digital computer memory.

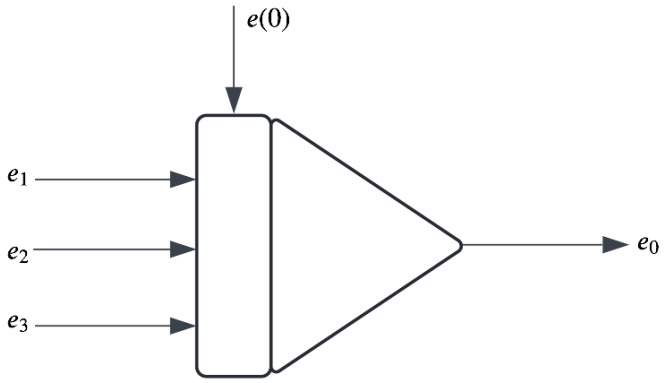

Figure 2.2: *An abstract representation of an integrator with three inputs. A fourth input, the initial condition $e(0)$, is also shown.*

The mathematical form of an integrator is as follows:

$$e_0 = -\int_0^t \sum_{i=1}^{n} a_i e_i \Delta t + e(0)$$

In this form, e_i is the input i, a_i the weight at that input, $e(0)$ the initial condition, and e_0 the current integrated value.

Integrators feature some necessary controls in all analog machines, which are generally physically operated. These controls are:

INITIAL Resets the integrator back to its initial state before any operations.

OPERATE Allows the integration of input values with respect to time.

HALT Halts integration, ensuring the integrator holds its current value. Note that, in practical applications, a halted integrator is prone to some noise accretion over time, and so should not be relied for accurate results over time.

Function generators

It is sometimes necessary to have an arbitrary function in analog computation. In some cases, these functions can be computed using some form of approximation or Taylor series, but in situations where they cannot be produced, the generation of the function may be implemented in hardware.

It is important to note that functions should be generated with programming of the analog computer; specific function generators are time-consuming and often single-use hardware. In the case of hybrid analog-digital computers, a digital computer could calculate the values of a function and communicate directly to the analog computer via a DAC to simulate a function generator.

Multiplication

Multipliers in analog machines have been hard to implement, but one widely used implementation follows the form:

$$xy = \frac{1}{4}((x+y)^2 - (x-y)^2)$$

This is known as a Quarter-Square multiplier, and can be implemented without unnecessary components such as a logarithm generator. Implementations of quarter-square multipliers require an inverted and a non-inverted input of both operands.

Free elements, Potentiometers

Free elements cover components such as capacitors, diodes, and resistors. These elements are employed as needed to extend capabilities of other components in the analog computer to yield specific functions — diodes enable one-way switches, resistors reduce by a certain amount, and capacitors can act as buffers.

Potentiometers, as expected, act as a set coefficient of a value between ± 1. This gives the ability to *scale* input values for certain values to be used later on when needed, which is crucial considering the limits of the machine unit on which an analog computer is based upon. Generally, potentionmeters are static, unlike a multiplier; the coefficient of a potentiometer is determined in the configuration or programming of an analog machine.
In classic analog computers, the accuracy of a potentiometer dial often deviates significantly with regards to the actual resistance of the potentiometer — this is due to factors such as the load behind the potentiometer. To this end, many analog computers feature additional modes of operation specific to setting up potentiometers, involving halting all integrators and connecting all inputs to ground to observe full load resistance. At this point, the potentiometers can be calibrated properly.

In modern analog computers utilizing operational amplifiers (a piece of hardware frequently used in all analog machines), certain components known as *impedance converters* can prevent this by unloading the potentiometer, reducing error sources to negligible current deviations. These potentiometers are known as *buffered* potentiometers.

Comparators

Comparators enable some logic outputs for analog signals by comparing two analog signals, e_0 and e_1, to the function $e_0 + e_1 > 0$. These logical outputs can connect to relays or switches in the analog computer and can allow for simpler digital logic systems throughout the computer, such as function switching.

Input & output

Most analog computers can be considered in the function of computing a graph, and as such, an oscilloscope or pen-plotter is the default output media for an analog computer, with an oscilloscope interfacing directly with electronic analog computers. Analog computers similarly operate with analog signal inputs; applications that usually would require the use of an ADC, such as sensors or actuators, are able to take full advantage of the analog computer's ability to operate on these analog signals directly, without the need for a conversion.

2.2 Hardware acceleration

Over the past few decades the speed of integrated chip development as per Moore's Law had been noted to slow down. First observed in 1965, it *loosely* held true over the next few decades, but had stated to slow as early as the mid-1970s as numerous huge developments had been delayed from release due to difficulties in production. The reason for this slowdown can be attributed to phenomena such as learning curve theory, but as of the early 21^{st} century, integrated electronics development for CPUs had hid a physical wall — quantum effects and other difficulties have prevented transistors from shrinking any more [11].

To this end, an increasing interest in computer hardware specialization has become the new method in order to keep up with the growing demands of computers. These have come in the form of notable hardware such as the GPU and FPGA. GPU hardware has seen a steady increase in power ever since its inception, primarily due to development focused on 'scaling outwards' — increasing the core count on GPUs to focus on parallelism as opposed to single-core performance. These hardware units are largely focused for computer graphics rendering, neural network applications, cryptography and cryptocurrency, and high-performance computing. Likewise, FPGAs have seen great use for low-level applications that require the specific strengths of power efficiency, low latency and good throughput [6].

Hardware acceleration is a growing field as CPU single-core power remains, physically, limited (although new techniques are regularly introduced to circumvent other physical limits, such as speculative execution). Hardware dedicated to machine learning applications from Google, Apple, and other competitors are on their way, while new, novel methods to handle contemporary problems such as video encoding are being developed [5]. As time goes on, more and more general-purpose software processes may be extrapolated to

hardware accelerated units to make use of the power and performance efficiency benefits they bring.

2.3 Modern Analog Computing

Currently, analog computers are seeing renewed interest as a next phase of development for neural network & AI applications, as the features of an analog computer - performance & energy efficiency, rate of computation - are well suited to neural network applications, as well as other matrix-based or data-processing systems such as Digital Signal Processors (DSPs) [8].

As mentioned earlier, analog computing (or, as it has been called in some modern implementations, approximate computing) has been demonstrated to perform operations at a same level of or greater than those performed on other systems such as FPGAs, GPUs and CPUs while maintaining a significant energy efficiency in watts, and a greater overall performance (FLOPs/J, or FPS/W) [4, 10]. The benefits granted are especially well suited to neural networks — the operations featured in neural network computation involve large amounts of parallel floating point operations on each layer of the network. Similarly, there have been applications of similar technologies in fields such as digital signal processing substitutes [8].

2.3.1 Components

The design of an in-computer analog array for the above purposes commonly uses a 2D crossbar array structure of analog computing sub-arrays [8, 10]. These sub-arrays generally have functionality applicable for data processing functions, such as weighting (coefficients), summation, and so on.

The 2D layout of such analog computers is essential to the functions it outputs. As these systems are used for signal processing and neural networks featuring connected layers, each 'cell' in an analog array depends greatly on the cells in the layers adjacent.

2.3.2 Findings

Across the board, analog computer architectures tend to perform better at their tasks than their digital contemporaries. Expanding on the points mentioned earlier on analog compared to digital:

- Analog computing chips in Neural Network operations demonstrate power draws of 1.475W compared to FPGAs, GPUs and CPUs on comparable workloads drawing 40W, 144W and 116W respectively [10]. Similar results are demonstrated on mobile devices, with novel IMAC interfaces yielding energy consumption improvements of 5-10% compared to baseline processors [4].

- Performance in terms of power can be measured in terms of FPS/W, with the tested analog computer measuring 6.214 FPS/W, while FPGAs, GPUs and CPUs measured 0.483, 0.091 & 0.023 FPS/W respectively [10]. This is a similar measurement to earlier FLOP/J ratios.

It is important to note that energy consumption and efficiency measures are subject to algorithm and workload, and may not be consistent across all applications.

Additionally, in in-memory analog computing arrays for CNN models, the accuracy difference compared to a baseline mobile processor was -0.9% and -0.27% difference across two different models, *LeNet-5* and *VGG* respectively [4]. Other applications of analog computing elements feature notable inaccuracies at less than 1.7% inaccuracy [13].

One caveat to these analog computing devices is summized in Amdahl's Law — as these analog computing systems tend to fill niche applications, and although they demonstrate great benefits to performance efficiencies, these gains are bounded by how often these systems are utilized. Integrated into a digital computer, the 'up-time' of these analog systems may still be small, depending on the use of the digital computer [4].

Chapter 3

Methodology

As mentioned in chapter 1.1, the goal of this project is as follows:

1. Design an abstract model of a programmable (configurable) analog-digital co-computer.
    (a) Implement a simulated version of the above co-computer in a library to allow use in digital computer code.
2. Identify software processes that may be performed on an analog co-computer, and, using the simulated model, implement these processes via an analog co-computer in a demonstration program(s).
    (a) Analyze aspects of the above implementations, including efficacy of working with the analog co-computer, predicted performance benefits, and other uses.

In the process of achieving these primary goals, the following questions will be considered:

1. What are the features & characteristics of an analog-digital co-computer?
    - What components of an analog computer are present, and how do they interlink?
    - How can this analog co-computer be configured?
    - Does this abstract model allow for, with proper configuration, operation achievable by a normal patch-panel analog computer?
2. How can a analog co-computer be effectively configured by a digital controller?
    - Assuming that any analog co-computer will have a limited capacity, is there an algorithmic method to configure the co-computer so that functions are mapped efficiently?
    - How does a software program interact with the analog co-computer?
3. What software processes should be offloaded to an analog co-computer?

- Defining a measure of 'analog-fitness' for software processes.
- Considering Amdahl's law - can larger scale programs be implemented.

4. What are the potential benefits of utilizing this analog co-computer?

These 4 questions should be answered as a byproduct of reaching the above mentioned goals.

The process of addressing all goals and questions will follow the following 3-step method.

3.1 Defining an abstract model

By analysing components of existing analog machines and how they're commonly interconnected, as well as referring to the structures of contemporary integrated analog computers, an **abstract** model of an analog co-computer will be designed. This model should meet the following set of criteria:

- Allows for programming by configuring what operations are performed and how those operations connect into one another.
- Contains all the essential components of an analog computer. If not all components are present (such as a multiplier), it should be possible to configure it to produce that components functionality.

Note that there may be more requirements that arise as the models are designed and checked. The models will be depicted in terms of theory & logic diagrams only — the physical hardware implementation of such a model is out-of-scope for this project.

If a model does not meet requirements, or has possibility to be improved upon, a new model will be designed and checked.

3.2 Implementing a simulation of the model

Once a satisfactory abstract model has been created, the next key step is to write a software simulation of the model. This will be done in a low-level language such as C or C++ to allow for higher speed simulation as well as use in many processes which may use these languages.

The simulation should be checked for correctness against physical analog computers, such as The Analog Thing on certain functions. If the simulation doesn't fall close to the physical analog computer, it will have to be rewritten, or the model redesigned to closer match a physical computer.

3.3 Identifying & implementing software processes in the simulated model

Given a specification of the model and the characteristics of analog computer, a form of 'analog fitness' can be designated that algorithms and software processes can be graded against. Ultimately, a number of software processes will be chosen to implement in the simulator so that the program can be practically run using the simulated analog co-computer.

3.4 Analysis

The scale of use of the analog model will be calculated for each implementation, and from there predicted benefits of using the analog model compared to running by default can be predicted. This process will include benchmarking of the whole programs in both analog and non-analog modes, as well as benchmarking of the non-analog code that was replaced specifically, which will be compared against the analog model computing the same values. Physical data on the performance of an analog model can be predicted based on known numbers as well as physical analog computers, such as The Analog Thing.

3.5 Summary

An abstract model will be proposed. This model will be 'verified' by analysing that it can be used to create any configuration or 'programming' possible on a normal analog computer. Once verified, it will be implemented in a software package and checked against a physical analog computer such as THAT to ensure that it performs functions the same as the physical analog computer. From there, the analysis of its benefits and use-cases can be performed.

Chapter 4

Model verification

To be completed in COMP4093, Thesis B.

Chapter 5

Results

To be completed in COMP4093, Thesis B.

5.1 Analysis

To be completed in COMP4093, Thesis B.

Chapter 6

Discussion & Future Work

To be completed in COMP4093, Thesis B.

Chapter 7

Conclusion

To be completed in COMP4093, Thesis B.

Chapter 8

Abbreviations

ADC	Analog-to-Digital Converter
AI	Artificial Intelligence
CNN	Convolutional Neural Network
CPU	Central Processing Unit
DAC	Digital-to-Analog Converter
DSP	Digital Signal Processor
FLOPs	Floating-point operations per second
FPGA	Field-programmable Gate Array
FPJ	FLOP-per-Joule
FPS	Frames-per-second
FPW	FLOPs-per-Watt
GPU	Graphical Processing Unit
IMAC	In-Memory Analog Computing
LLM	Large-Language Model

Bibliography

[1] "X-15 E-6910: X-15 simulator." [Online]. Available: https://www.dfrc.nasa.gov/Gallery/Photo/X-15/HTML/E-6910.html

[2] "anabrid/the-analog-thing," May 2024, original-date: 2022-03-14T09:03:52Z. [Online]. Available: https://github.com/anabrid/the-analog-thing

[3] C. Care, "Introduction: Analogue Computers in the History of Computing," in *Technology for Modelling: Electrical Analogies, Engineering Practice, and the Development of Analogue Computing*, C. Care, Ed. London: Springer, 2010, pp. 3–16. [Online]. Available: https://doi.org/10.1007/978-1-84882-948-0_1

[4] M. Elbtity, A. Singh, B. Reidy, X. Guo, and R. Zand, "An In-Memory Analog Computing Co-Processor for Energy-Efficient CNN Inference on Mobile Devices," in *2021 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, Jul. 2021, pp. 188–193, arXiv:2105.13904 [cs]. [Online]. Available: http://arxiv.org/abs/2105.13904

[5] R. Husemann, A. A. Susin, and V. Roesler, "Optimized Solution to Accelerate in Hardware an Intra H.264/SVC Video Encoder," *IEEE Micro*, vol. 38, no. 6, pp. 8–17, Nov. 2018, conference Name: IEEE Micro. [Online]. Available: https://ieeexplore.ieee.org/document/8536428/

[6] W.-m. Hwu and S. Patel, "Accelerator Architectures —A Ten-Year Retrospective," *IEEE Micro*, vol. 38, no. 6, pp. 56–62, Nov. 2018, conference Name: IEEE Micro. [Online]. Available: https://ieeexplore.ieee.org/document/8585394/

[7] M. Kim and Y. S. Shao, "Hardware Acceleration," *IEEE Micro*, vol. 38, no. 6, pp. 6–7, Nov. 2018, conference Name: IEEE Micro. [Online]. Available: https://ieeexplore.ieee.org/document/8585396

[8] M. R. Kucic, "Analog computing arrays," Ph.D., Georgia Institute of Technology, United States – Georgia, iSBN: 9780496155842. [Online]. Available: https://www.proquest.com/docview/305184996/abstract/819F09156DF340D7PQ/1

[9] S. Köppel, B. Ulmann, L. Heimann, and D. Killat, "Using analog computers in today's largest computational challenges," in *Advances in Radio Science*, vol. 19. Copernicus GmbH, Dec. 2021, pp. 105–116, iSSN: 1684-9965 Issue: D. [Online]. Available: https://ars.copernicus.org/articles/19/105/2021/

[10] B. Li, P. Gu, Y. Shan, Y. Wang, Y. Chen, and H. Yang, "RRAM-Based Analog Approximate Computing," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 12, pp. 1905–1917, Dec. 2015, conference Name: IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems. [Online]. Available: https://ieeexplore.ieee.org/document/7123597

[11] C. A. Mack, "Fifty Years of Moore's Law," *IEEE Transactions on Semiconductor Manufacturing*, vol. 24, no. 2, pp. 202–207, May 2011, conference Name: IEEE Transactions on Semiconductor Manufacturing. [Online]. Available: https://ieeexplore.ieee.org/document/5696765

[12] B. Ulmann, "Analog and Hybrid Computer Programming," in *Analog and Hybrid Computer Programming*. De Gruyter Oldenbourg, May 2023. [Online]. Available: https://www.degruyter.com/document/doi/10.1515/9783110787733/html

[13] R. Zhang, N. Uetake, T. Nakada, and Y. Nakashima, "Design of Programmable Analog Calculation Unit by Implementing Support Vector Regression for Approximate Computing," *IEEE Micro*, vol. 38, no. 6, pp. 73–82, Nov. 2018, conference Name: IEEE Micro. [Online]. Available: https://ieeexplore.ieee.org/document/8486976/