**NTNU – Trondheim**
Norwegian University of
Science and Technology

# Isolating the Black Box

Exploring the Effects of a Material Interface
in Evolution in Materio

Erik Lothe

Master of Science in Informatics
Submission date: June 2015
Supervisor: Gunnar Tufte, IDI

Norwegian University of Science and Technology
Department of Computer and Information Science

# Problem description

This project work will be part of the EU-funded Future Emerging Technologies initiative. In this research project we try to exploit computational properties of unconventional materials (materials usually not considered as a computational substrate). Such materials may offer computation at extreme low cost and may also enable us to do computation that is hard (or impossible) on a Von Neumann stored program machine. Currently we explore possible computational properties of carbon nano tubes.

In this sub-project the goal is to expand the monitoring capability by designing additional interface circuits. The main objective is to separate the measurement hardware from material samples, i.e. the computing substrate, as to be able to collect data for the purpose of generating models of the computing nano structures. Project tasks include:

- Basic analogue and digital design.
- PCB design.
- Interfacing to existing hardware (HDL-design).
- Software to support new functionality.
- Initial experiments.

# Abstract

Evolution in Materio (EiM) is the concept of exploiting intrinsic properties of various materials, or "black boxes", to perform computation. Motivation for EiM is multitudinous; new paradigms for computing, increased performance, and basic research on what computation is or can be. The work herein is part of the NASCENCE project, which aims at exploring nanoscale devices for computation and, further, how such information processing devices can emerge from bottom-up design processes, e.g. artificial evolution. In EiM research, instability and lack of determinism has been an ongoing problem regarding the evolution of useful computation. Meanwhile there has also been a lack of a clear, distinct border between the mechanism that exploits the material and the material itself. This raises questions like; where does computation arise? In the material, in the interface to the environment, or a combination?

In this thesis we have investigated the effect of drawing such a border by designing and implementing a material interface named optowall. This interface electrically isolates the material by using opto-isolators to transmit all signals by light instead of by an electrical connection. An experimental approach has been taken to develop methods to define properties in the material such as stability, functionality, and the nature of instability. By developing software that extracts the values of these properties, we have compared them in different configurations of the interface and found that the method of delivering the signal into the material has a significant effect on the computational properties. Hence, the impact also influence the search landscape for any evolutionary algorithm. We have also discovered that non-linear functionality can be exploited by the adjustment of current. Furthermore we have shown that the potential differences between concurrent input signals have a significant effect on stability and functionality.

# Sammendrag

Evolution in Materio (EiM) er et konsept der man utnytter fysiske egenskaper i forskjellige materialer for å utføre beregninger. Forskning på EiM kan motiveres av oppdagelse av nye komputasjonsparadigmer, økt ytelse, og forståelse av hva komputasjon kan være. Arbeidet i denne avhandlingen er en del av forskningsprosjektet NASCENCE, som etterstreber å utforske enheter i nanoskalaen for komputasjon og videre hvordan slike informasjonsbehandlende enheter kan oppstå fra designprosesser som går fra bunnen og oppover, som for eksempel kunstig evolusjon. I EiM-forskning har ustabilitet og manglende determinisme vært et gjennomgående problem vedrørende evolusjon av anvendbar komputasjon. Samtidig har det vært mangel på en klar og distinkt grense mellom mekanismen som utnytter materialet og selve materialet. Dette vekker spørsmål som; hvor oppstår komputasjonen? I selve materialet, grensesnittet mellom mekanismen som utnytter materialet, eller en kombinasjon?

I denne avhandlingen har vi undersøkt effekten av å definere en slik grense ved å designe og implementere et grensesnitt for materialet kalt optowall. Dette grensesnittet isolerer materialet med hensyn til elektrisitet, og dette oppnås ved å bruke optokoblere til å overføre signaler med lys istedet for elektriske signal. En eksperimentell framgangsmåte har blitt brukt for å utvikle metoder for å definere egenskaper i materialet slik som stabilitet, funksjonalitet, og mønstre ved ustabilitet. Vi har utviklet programvare som kan hente ut verdiene til egenskapene, og har sammelignet disse i forskjellige konfigurasjoner av grensesnittet. Fra dette har vi oppdaget at metoden for å levere et signal inn til materialet har en signifikant påvirkning på de komputasjonelle egenskapene. Dette har videre en påvirkning på søkerommet til en evolusjonær algoritme. Vi har også oppdaget at ikke-lineær funksjonalitet kan utnyttes ved hjelp av justeringer i mengden elektrisk strøm (I). Videre har vi vist at spenningsforskjellen mellom samtidige input-signal har en signifikant påvirkning på stabilitet og funksjonalitet.

# Acknowledgements

First of all, I would like to thank my supervisor **Gunnar Tufte**. I am sincerely grateful for his interest and investment in my thesis, and his advice on both the research direction and technical specifics has been invaluable. During my time at NTNU, his enthusiasm and approachableness has greatly contributed to my academic development and passion for computer science.

I would also like to thank **Odd Rune Strømmen Lykkebø** and **Stefano Nichele** for all their help in getting Mecobo to work, giving me feedback on my report, guiding me on some design choices, and helping me putting in the production order for the optoamp PCB. I also want to thank **Kai Torgeir Dragland** for lending me equipment and giving me advice for photographing my PCBs, and my friends **Audun Gevelt**, **Tom Glover**, and **Bjørn Åge Tungesvik** for giving me feedback on my report.

Lastly, I would like to thank my girlfriend **Beate Baier Biribakken** for both her emotional and academic support.

# Contents


**1 Introduction** **17**

**2 Background** **21**
2.1 Restrictive computers . . . . . 21
2.2 Computational potential of unknown components . . . . . 22
2.3 Evolution by genetic algorithms . . . . . 24
2.4 In simulo vs. in materio evolution . . . . . 26

**3 Previous work** **29**
3.1 Gordon Pask's electrochemical devices . . . . . 29
3.2 Thompson's evolvable FPGA . . . . . 31
3.3 Layzell's "test bed" for intrinsic hardware evolution . . . . . 33
3.4 Harding and Miller's liquid crystal tone discriminator . . . . . 36
3.5 NASCENCE's EiM platform . . . . . 39

**4 Methodology** **43**
4.1 Requirements of interface . . . . . 43
4.2 The Mecobo platform . . . . . 44
4.3 Circuit design of material interface . . . . . 45
4.4 Hardware realization of material interface . . . . . 48
4.5 Integrating the optowall daughterboard with Mecobo . . . . . 52
4.6 Problems arising from the use of opto-isolators . . . . . 54
4.7 Determinism of experimental tools . . . . . 59

**5 Experiments** **61**
5.1 Common experimental setups . . . . . 61
5.2 Exhaustive sweep experiment . . . . . 63
5.3 Stable exhaustive sweep experiment . . . . . 66
5.4 Comparing information content of material input waves . . . . . 68
5.5 Qualitative stability rating . . . . . 70
5.6 Quantitative stability rating . . . . . 73
5.7 Comparison of stability and input vectors . . . . . 78
5.8 Genetic search for logic gates . . . . . 80

**6 Results and Discussion** **83**
6.1 Method of signal delivery . . . . . 83
6.2 Functionality caused by adjustment of current . . . . . 84
6.3 Relationship between stability and functionality . . . . . 84
6.4 Finding the "sweet spot" . . . . . 85
6.5 Future work . . . . . 86

**7 Conclusion** **89**

**Appendices** **93**

**A PCB documents** **93**

**B Pin mappings** **105**

**C Software as experimental tools** **107**
C.1 Optoprober . . . . . 107
C.2 Mecoboprober . . . . . 107

**D Files** **111**
D.1 opto repository . . . . . 111
D.2 NASCENCE/mecobo repository . . . . . 111

# List of Figures


1.1 Evolution in Materio[1] . . . 18
1.2 Material slide used by NASCENCE . . . 19

2.1 A black box with seven I/O wires . . . 23
2.2 Two trajectories of the logistic map, recreated from Mitchell[2] . . . 26

3.1 Pask's electrochemical device [3] . . . 30
3.2 Thompson's experimental arrangement[4] . . . 31
3.3 GA progress of Thompson's FPGA experiment[4] . . . 32
3.4 A simplified representation of the Evolvable Motherboard[5] . . . 33
3.5 Circuit diagrams for first experiment[5] . . . 34
3.6 Circuits instantiated for the first experiment[5] . . . 35
3.7 Schematic of LCEM[6] . . . 37
3.8 Layers in an LCD[6] . . . 37
3.9 Evolution of a non-linear response[6] . . . 38
3.10 Tone discriminator response[6] . . . 38
3.11 All possible gate output sums [7] . . . 40
3.12 Experimental setup for genetic search[7] . . . 41

4.1 Mecobo . . . 45
4.2 Mecobo with optowall as daughterboard . . . 46
4.3 Material interface circuit . . . 47
4.4 Optowall . . . 49
4.5 Power header . . . 50
4.6 Mecobo power connectors . . . 50
4.7 Optowall with the optoamp extension board . . . 51
4.8 Waves translating through opto-in . . . 53
4.9 Effects of opto-isolator on (the unused) pin 2 . . . 55
4.10 opto-out resistance . . . 56
4.11 Material-side output amplifier effects . . . 57
4.12 Exhaustive sweeps with amplifiers on input pins . . . 58
4.13 Opto-isolator's effects on square waves of different frequencies . . . 59

5.1 Gate output sums on material with 1.25% consentration (B08S02) . . . 65

5.2 Gate output sums on material with 5% consentration (B08S04) . . . 66
5.3 Stable gate output sums on material with 1.25% consentration (B08S02) 67
5.4 Information difference of material input square wave . . . . . 69
5.5 DFT of software modulated square wave . . . . . 70
5.6 Stablity rating algorithm examples . . . . . 71
5.7 Adjustment of gain . . . . . 72
5.8 Stablity rating . . . . . 73
5.9 Visualization of $d$ function . . . . . 76
5.10 Effects of gain . . . . . 77
5.11 Distribution of m(v) for all $2^{15}$ input vectors . . . . . 78
5.12 Relationship between mean of input vector and stability . . . . . 79
5.13 Evolution of fitness . . . . . 82

6.1 Software gain control . . . . . 87

C.1 Software diagram . . . . . 108

# List of Tables

2.1 Material function example . . . . . 24
2.2 Reproduction by genetic crossover . . . . . 25

3.1 Gate sum mapping of XOR[7] . . . . . 40

4.1 Relationship between input and output of the opto-in opto-isolator . 48

5.1 Material properties . . . . . 63
5.2 Gate sum extraction with input pins 0 and 14 . . . . . 64
5.3 Score weighting for different input vectors . . . . . 81

B.1 Pin mappings . . . . . 106

# Abbreviations

**CAP** Configurable Analog Processor

**DFT** Discrete Fourier Transform

**EiM** Evolution in Materio

**EM** Evolvable Motherboard

**FPGA** Field Programmable Gate Array

**FPMA** Field Programmable Matter Array

**GA** Genetic Algorithm

**IHE** Intrinsic Hardware Evolution

**LC** Liquid Crystal

**LCEM** Liquid Crystal Evolvable Motherboard

**PCB** Printed Circuit Board

**QLSR** Qualitative Stability Rating

**QNSR** Quantitative Stability Rating

# Chapter 1

# Introduction

This thesis is part of NAnoSCale Engineering for Novel Computation using Evolution (NASCENCE). The aim of the NASCENCE project is "to model, understand and exploit the behaviour of evolving nanosystems (e.g. networks of nanoparticles, carbon nanotubes or films of graphene) with the long term goal to build information processing devices exploiting these architectures without reproducing individual components."[1]

The conventional Turing machine achieves its universal computation through the use of symbols, and its typical Von Neumann implementation realizes this by enforcing several *layers of abstraction* on top of the physical components. This also means that, at every layer, *restrictions* are applied to the physical capabilities of the electronics, and potential computational properties are lost. Moore's law[8], essentially prophesying the continued improvement of the conventional computers, relies on the continuing miniaturization of electronics. Because of fundamental physical limitations such as the presence of fundamental building blocks (electronics cannot be smaller than atoms), and dissipation of energy (the amount of heat generated per area will ultimately be impossible to dissipate), Moore's law is likely to break down in the near future[9].

In the research of NASCENCE, the goal is to utilize Evolution in Materio (EiM)[1] to evolve unconventional computers with as few restrictions as possible, thus "moving beyond the Turing/Von Neumann concept of computing"[7]. As illustrated by figure 1.1, this works roughly by applying signals into a computational material, e.g. an FPGA[4], liquid crystal[6], or carbon nanotubes[7]; reading some output signal; and testing this signal for a desired response. A Genetic Algorithm (GA)[10] can repeat this process while continually altering the input signals to better meet the desired output response. In this scheme, evolution works directly with the physical medium and is able to discover and exploit *intrinsic* computational properties in it. Because these properties are highly complicated, and their exploitation often appear counterintuitive and bizarre[4], they are practically impossible for a human designer to recognize and exploit.

[1] NASCENCE web page: www.nascence.eu

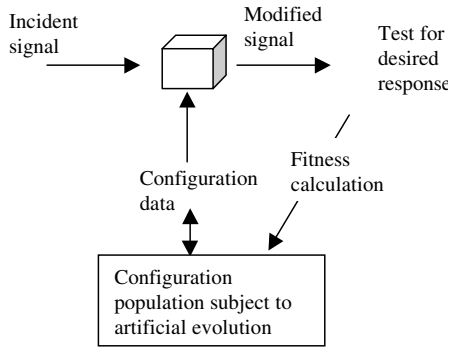



Figure 1.1: Evolution in Materio[1]

There are several advantages to this approach. For one, removing the restrictions of the conventional computers and instead exploiting the actual physics may allow for potentially huge improvements in performance. Because systems are evolved at nanoscale by exploration and explotation, we have the advantages of nanoscale information processing devices, but at a much lower cost. Also, by allowing evolution to independently work out the details of how to solve a problem, the human requirement may be reduced to producing a behavioural specification of the system. As this might require significantly less knowledge of electronics, it can potentially open up the field of hardware design to non-specialists[5].

A hardware and software platform called Mecobo has been developed by NASCENCE for this kind of work[7, 11]. It consists, roughly, of a PC, microcontroller, Field Programmable Gate Array (FPGA), and a computational material. An example of such a computational material is shown in figure 1.2. In this figure, the small round black container holds a sample of single walled carbon nanotubes. An electrode array connects this material to wires, or *material pins*, by a total of 18 connectors which allows electrical signals to be applied through it at different positions. A GA runs on the PC, and "probes" this material with a candidate program and some sample data, both encoded as electrical signals such as waves and constant voltages. Such an action is performed by configuring the rest of the Mecobo platform to generate the given set of signals onto the given set of material pins, and reading a response on one or more given pins. From the perspective of the GA, every part of this process can, like the material itself, be viewed as a black box.

**As the GA is free to exploit *any* properties it may discover, it may also be free to exploit properties of the hardware in between itself and the material.**

The only requirement of electronics is that they conform to their abstract model, so any imperfections that does not invalidate the model are acceptable. As no electronics are perfect, certain insignificant anomalies on the electrical characteristics might occur after certain sequences of events. As the material is on the same electric net as the rest of the system, such anomalies would then also be present in the material. In the case of *known* electronics, such as digital circuits, this is accept-

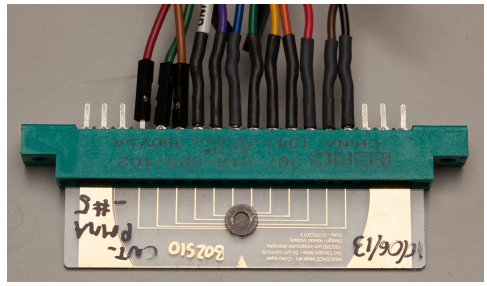


Figure 1.2: Material slide used by NASCENCE

able because when the imperfections are kept below a certain level, the abstract model of the circuit holds. However, in the case of *unknown* electronics, such as the material, there is no such model, and we cannot know for sure whether small anomalies will affect the emergent behaviour.

The GA, blind to the inner workings of the platform, has no way of distinguishing such effects from the effects of the material alone. For instance, if the GA configures Mecobo to apply a set of signals on a set of pins and reads a response, that specific configuration may cause the impedance on the pins to be different than in other configurations. Such effects, though negligible in the digital model, may cause chain reactions in the material resulting in a different outcome. Because of this, the GA will end up exploiting properties of Mecobo, and thus the border between Mecobo and the material sample becomes unclear.

**The main research goal of this thesis is to study the effect that the method of signal delivery has on the computational properties of the material.** To do this, we have electrically isolated the material from the complex circuitry of Mecobo, by using opto-isolators to transmit all signals by light instead of by a direct electrical connection. In the broader sense, this denies the GA access to exploit any other properties than those it discovers in the material. The research has taken an experimental approach, and thus the specific research questions has been continually formulated during the course of the research process. The three research questions resulting from this process is presented below.

As Mecobo, and several other EiM approaches, works by routing externally generated signals into a seemingly passive material, one might view the material as a passive component. Our most fundamental research question is whether this view holds: Does the properties of the circuitry that generates the signals affect the properties observed in the material?

Traditionally, the input signals have consisted of three main variables: Amplitude (voltage), frequency, and duty cycle. Are there other kinds of electrical properties that can cause interesting behaviour? Specifically, can the adjustment of *current* trigger non-linear functionality?

In EiM, behaviour in the material can sometimes be volatile and unstable, a fact that carries obvious implications for applicability. What is the relationship between functionality and stability? Is there a pattern to how stability affects

functionality and vice versa? Also, is there a pattern to how the instability arises? What is the *nature* of instability?

### Report structure

The remainder of this report is structured as follows. In chapter 2 we will give a general introduction into the domain knowledge on which this thesis rests. We will then, in chapter 3, highlight some key research efforts that have driven the field forward. Following, in chapter 4, we will document the design, implementation, and functionality of the tools we have developed to attempt to answer the questions raised above. Chapter 5 presents our initial experiments and their immediate results, before the meaning of these results are discussed in chapter 6. Finally, chapter 7 concludes the work.

For replication or closer inspection of the experimental process, additional information is included in the appendices. In appendix A, PCB documents are included, containing schematics, bill of materials, and gerber files. A summary of pin mappings on the PCBs are included in appendix B. The software developed and used for the experiments is documented in appendix C. Finally, appendix D describes the location of all files associated with the work of this thesis, e.g. source code and experiment data.

# Chapter 2

# Background

This chapter introduces the domain knowledge relevant to this thesis. We begin by reflecting upon the central design philosophy of conventional computers, the abstraction layers, and its restrictive effect in section 2.1. Second, in section 2.2, we will examine the alternative Configurable Analog Processor (CAP) and the computational potentials it might hold. As a means of practical CAP utilization, the Genetic Algorithm (GA) is introduced in section 2.3. Finally, in section 2.4, we join the GA with the CAP and discuss its role as evolution *in materio*.

## 2.1 Restrictive computers

According to Koza[12], most achievements of science have followed the seven principles of correctness, consistency, justifiability, certainty, orderliness, parsimony, and decisiveness. In the engineering disciplines, and specifically the building of computers, this translates to the top-down design process. Thompson[4] generalizes this approach as the following strategy: "(1) Break the system into smaller parts that can be understood individually. (2) Restrict the interactions between these parts so *that* can be understood. (3) Apply 1 and 2 hierarchically, allowing design at increasing levels of abstraction."

The conventional computers works by manipulating symbols[13], a principle which stems from the Turing Machine[14]. The fundamental symbols are the binary `HIGH` and `LOW` which are each defined to a continuous range of voltages that are, respectively, "high", e.g. 4.0V to 5.5V, or "low", e.g. -0.5V to 1.0V[15]. These symbols often represent binary digits, i.e. `1` and `0`, but can also have other meanings such as `TRUE` and `FALSE`. These fundamental symbols can be seen as an abstraction of the electrical signal.

In practice, the abstraction from electric signal to binary symbol is performed by the silicon transistor[1]. Its behaviour is restricted to a simple *model of operation*, effectively hiding most of the complexity of its underlaying electrical principles. In modern times, the silicon transistor functions as the fundamental building block

[1] https://en.wikipedia.org/wiki/Transistor#Transistor_as_a_switch

of conventional computers. It is used to implement logic gates[2], which in turn are used to implement higher level components of the computer[15].

The key to this top-down process is that all components of the computer, be it a single transistor or a high level component such as a CPU, must completely conform to a simplifying model, and any interaction between components must exclusively occur within the restrictions of said models[13]. This allows higher level-components to be specified not by the means of complicated physics, but by simplified models of operations of its constituent components. Because of the existence of transistors, for instance, the behavioural model of a logic gate need not be specified based on complex electrical principles, but rather on the simple transistor model. The logic gates can then be classified as a higher layer of abstraction.

The logic gates, and indeed every layer of abstraction, will hide complexity from the next upper level. This severely limits the complexity of the individual components and is a key idea for humans to be able to design computers and other complex machines. However, the act of hiding complexity is synonymously an act of *restriction*: Every time an abstraction is applied, something potentially useful is lost. In the case of transistors and logic gates, this loss concerns the many physical/electrical properties that are made inaccessible during the process of abstraction. From this it follows that of all the ways that computation can physically occur, *conventional* computers might be utilizing only a very small fraction[13].

In NASCENCE, the goal is to evolve computers with as few restrictions as possible, so as to be able to exploit as many physical processes for computation as possible. Such machines are known as *unconventional* computers and has no ties to any top-down architectures such as the Von Neumann architecture. The key to achieving this, is to surrender the requirement of understanding the inner workings of the computer. This is made possible by a technique known as evolution in materio.

## 2.2 Computational potential of unknown components

A core idea behind Evolution in Materio (EiM) is that computation is a vastly wider phenomenon than what goes on inside a Von Neumann stored-program computer. A lot of physical processes in nature can be viewed as computation, e.g. the rolling movement of boiling water[16]. EiM is the concept of using evolution to exploit the intrinsic properties of materials, or "computational mediums", to do computation, where neither the structure nor computational properties of the material needs to be known in advance[1]. This way, we can exploit natural physical processes to do useful computation.

Imagine *some* sort of computational medium, a black box, with several wires that we can either apply to, or read from, an electrical signal. It may be as shown in figure 2.1, where the inner structure is complicated and difficult to understand. If we were to apply a signal as input on some of the wires, we would be able to

[2] https://en.wikibooks.org/wiki/Foundations_of_Computer_Science/Computing_Machinery

measure some other signal as output on the remaining wires. By being flexible on the choice of input and output wires, and also the range of different signals to apply, the black box can be viewed as a function, albeit an arbitrary one. Miller and Downing calls such a device a Configurable Analog Processor (CAP)[1]. As long as there is enough electrical conductivity for a voltage to propagate through it, we would be able to take measures to domesticate this function by applying a state onto it through some of its input wires. The problem of creating a useful function from this black box would then be reduced to discovering a state that, when applied, makes the relationship between some chosen input and output wires match that of the desired function[13].

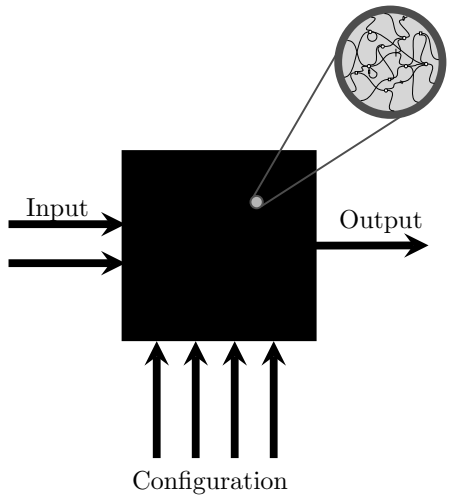



Figure 2.1: A black box with seven I/O wires

The simplest example of such a domestication might be that of finding a basic logic function, such as an OR function, through a digital time-independent static configuration[7]. This means that we divide the input into two groups: The static configuration, and the function input. Say, for instance, that the black box has seven wires, as illustrated in figure 2.1. Any signals that are either sent as input or read as output from the black box can be encoded as a bit string, where each bit represents, for one wire, either high voltage, "1", or low voltage, "0". By some means we discover that when we apply a certain digital bit string, for example "1010", to four of the wires (the *configuration*), the relationship between the remaining two input wires and a single output wire matches that of an OR. This scenario is summarized in table 2.1. As long as this configuration is applied, we can use the CAP as an OR component.

This principle can be scaled up to more wires and more types of signals, such as waves, to produce more complicated functions. The problem of domesticating unknown materials, or CAPs, can then be reduced to discovering a combination of all such variables that, when imposed on the computational medium, causes it to

| Configuration | Function input | Function output |
|---|---|---|
| 1010 | 00 | 0 |
| 1010 | 01 | 1 |
| 1010 | 10 | 1 |
| 1010 | 11 | 1 |

Table 2.1: Material function example

behave as desired.

## 2.3 Evolution by genetic algorithms

As the search space of states would consist of every possible permutation of all variables, it can become extremely large and an efficient heuristic search algorithm is essential. In mathematical terms, traditional methods of optimizing a function include calculus-based and enumerative methods. They are, however, unsuitable for noisy and discontinuous functions such as our black box[10]. Calculus-based methods can find local extrema fairly easily, but falls short when there are many local extrema. Additionally, they also have strict requirements about the function at hand, such as continuity and derivative existence. Enumerative schemes, where every point in the search space is considered one at a time, are not limited by function restrictions or local extrema. They are, however, very inefficient and of little practical use for large search spaces.

The Genetic Algorithm (GA) is a heuristic search algorithm formalized by David Goldberg[10], and based on ideas by John Holland[17]. It uses evolution and genetics to efficiently find approximations to global maxima in functions that can be multidimensional, discontinuous, and highly complex. Put in less mathematical terms, a GA *evolves* solutions to a given problem where the solutions are represented by *structures*, analogous to nature's genotypes[10]. Goldberg describes the GA as "nature's favorite search algorithm" and describes several practical uses of GAs such as facial recognition, design optimization, and currency trading prediction[18]. He states that, unlike many heuristic search algorithms, GAs are "broadly competent", "remarkably noise tolerant", and have a "clean interface" allowing them to be easily re-purposed for many different problems. Because of qualities such as these, the GA and evolutionary algorithms in general, are appropriate search algorithms for EiM[1, 13, 4, 7].

In the black box example of section 2.2, a structure would in the simplest case be the configuration bit string, such as "1010". In a case where the solution also includes the pin configuration, i.e. which connections are input, output and configuration, this would need to be specified in the structure as well. This could, for example, be described in the first $x + y$ genes, or *features* in GA terms, of the bit string structure. The first $x$ genes could then hold the pin numbers of the $x$ input pins and the following $y$ genes would similarly hold the $y$ output pins. The remaining pins could then be assumed to be configuration pins. A complete

| | |
|---|---|
| First parent | xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx |
| Second parent | yyyyyyyyyyyyyyyyyyyyyyyyyyyyyy |
| First offspring | xxxxxxxxyyyyyyyyyyyyyyyyyyyyyy |
| Second offspring | yyyyyyyyxxxxxxxxxxxxxxxxxxxxxx |

Table 2.2: Reproduction by genetic crossover

structure like this would represent one solution, or *individual*.

```
1  population = generate random population of size n
2  while determination criteria not fulfilled:
3      for each individual in population:
4          measure fitness of individual
5      while size of next generation < n:
6          parent1   = select from population
7          parent2   = select from population
8          offspring = crossover(parent1, parent2)
9          if chance(): mutate offspring
10         append offspring to next generation
11     population = next generation
```

Listing 2.1: Pseudocode for a genetic algorithm

Genetic algorithms have many variations, and an example GA is shown in listing 2.1[10]. The algorithm works with succeeding generations of a population, one at a time. The first generation is generated by random (line 1), and every succeeding generation is generated by three main phases: Fitness, selection and crossover. The fitness test (line 3 and 4) is performed on every individual and quantifies how well it performed in accordance to the desired functionality: The better an individual works, the higher fitness it is given. In the next phase, individuals are selected for reproduction (line 6 and 7). There are several ways to do this, and one way is selecting by probability based on fitness, meaning that individuals with high fitness are more *likely* to be selected. Goldberg visualizes this process as the spin of a roulette wheel, where every individual is given a slot on the wheel sized in proportion to its fitness. This is often combined with elitism, where only the best individuals are selected. The crossover phase (line 8) is the process of creating one or two offspring from two selected parent individuals. In a single-point crossover, a random crossover point is generated somewhere inside the length of the structure. The offspring are then created from a combination of the genes from one side of the crossover point from the first parent and the genes from the other side of the crossover point from the second parent. This process can create two offspring where each offspring contains an inverse selection from each parent, as illustrated in table 2.2. Offspring can also, once in a while, be introduced to small random mutations to keep the gene pool diverse (line 9). Individuals are repeatedly added to the next generation like this until the generation size becomes `n` (line 10 and 5). This process is repeated for either a fixed number of generations or until some termination criteria is fulfilled (line 11). Over the course of its execution, this process should evolve individuals with increasingly high fitness.

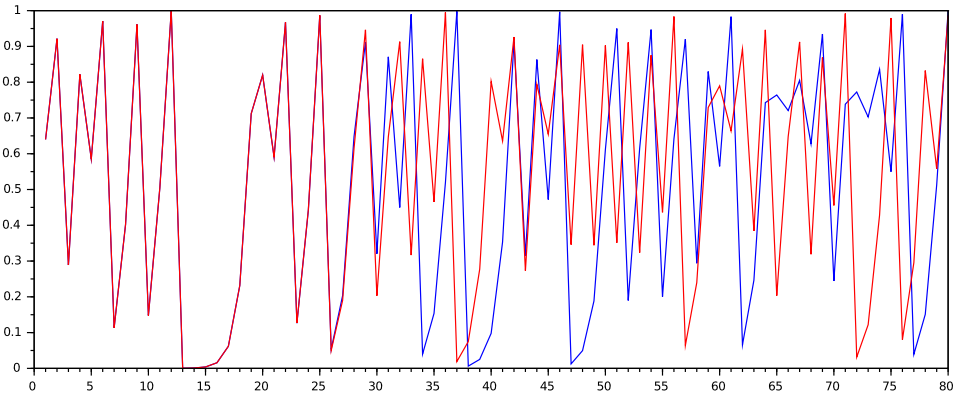



Figure 2.2: Two trajectories of the logistic map, recreated from Mitchell[2]

## 2.4 In simulo vs. in materio evolution

For practical reasons, GAs are normally confined to a simulated reality. For instance, GAs in combination with simulators have been used to evolve antennas for a NASA satellite[19]. It is hard to imagine how a fitness function in such a GA could evaluate solutions to this in a timely and automated manner *without* simulators. It would not only have to physically construct the antenna, but also launch it into space. On top of this, this fitness test would have to be repeated for a number of times proportional to the population size multiplied by the number of generations[10], a number which could easily rise into the thousands. For a GA of this sort to be of any practical use, its fitness function must use a simulator. This is known as evolution in simulo, because the evolution has no reach beyond its simulated environment. If there were any significant shortcomings of the simulator, the GA would have no way of accounting for it; it is ultimately bound by the accuracy of the simulator[1].

The in simulo evolved satellite antenna proved competitive to human designs[19], which shows that a simulator *can* be sufficiently accurate. Because of this, we can assume that there were no significant unknown aspects of the physical realization that needed to be taken into account in order to evolve the antenna. In general, however, we cannot necessarily make such an assumption because some aspects of the real world are inherently chaotic.

### 2.4.1 Chaotic systems

A simple mathematical model, known as the logistic map, can be used to demonstrate how complex and chaotic behaviour can arise from very simple non-linear dynamical equations[20]. This model is shown in equation 2.1.

$$x_{t+1} = ax_t(1 - x_t) \tag{2.1}$$

The primary intent of the model is to calculate population growth. In equation 2.1, $x$ represents the fraction of carrying capacity for a population of individuals in an environment. $x_t$ means the current value of $x$, and $x_{t+1}$ means the next value of $x$, which means that the equation can be used to see the growth of a population over time. The constant $a$ describes both birth- and death rate in the population. By ignoring this application and simply focusing on the equation itself, we can make a sound argument against the Newtonian clockwork universe[2]. With low values of $a$, the population growth is simple and predictable, as the simplicity of the equation would suggest. However, as the value of $a$ increases, the population growth becomes highly chaotic. Figure 2.2 shows two trajectories of the logistic map with $a = 4$, where the X axis represents time. The only difference between the red and the blue plot is that the initial conditions $x_0$ differs by $10^{-10}$, i.e. the blue has $x_0 = 0.2000000000$ and the red has $x_0 = 0.2000000001$. We see that they follow a similar path for a while, but eventually diverges into very different trajectories. This is an example of sensitive dependence on initial conditions. In such a chaotic system, "if there is any uncertainty in the initial condition $x_0$, there exists a time beyond which the future value cannot be predicted". This means that "a perfect prediction is impossible not only in practice but also *in principle*, since we can never know $x_0$ to infinitely many decimal places"[2]. Weather is an example of such a complex system, and the common failures of long term weather prediction further illustrates that some aspects of the real world are in fact impossible to accurately model. The computational mediums of EiM are also thought to be in this category.

### 2.4.2 Opening the black box

In the 1940s, the first generation cybernetic movement emerged in Britain and the USA. In the British branch, many had background in medicine or psychiatry[13], and the main topics were directed at adaptation and the brain. Many of the cyberneticians viewed the brain as a "black box", that could be opened and ultimately understood. Second generation British cyberneticians Gordon Pask and Stafford Beer had a different view. Lacking the strong connection to psychiatry as many of their fellows, their motivation was to explore computation rather than to understand the brain[13].

In his 1961 set theoretic formulation of the brain model, Beer describes an aspect of the brain called passive sensibility[21]. Beer defined this as "a collection of inputs which inform the brain about the state of the world, which is in turn defined as the state of the whole organism in relation to its environment." Thus, in this aspect, the brain is a state machine whose state is defined directly and passively by the combination of its inputs. Beer suggested that the total amount of sensory configurations in a brain would be $2^{2^n}$, where $n$ is the total amount of sensory inputs. Following Beer's presentation of his theory at the University of Illinois Symposium on Self-Organization in 1961, the enormity of this number was the subject of discussion. It was argued that when $n > 6$, it would be absurd to try and compute the functions of this brain. In Beer's view, this was correct but irrelevant: There was no need to try and open and understand the brain, it

could remain as a black box. The model must be constrained, but not by "the ignorant designer who has no idea what life in the future will be like." Instead, the model should be constrained and exploited by its own experiences and epigenetic landscape.

In retrospect, Beer's views are very similar to those of modern EiM researchers: In EiM, the focus is not on understanding the CAP itself, but rather training it to restrict its complexity in a way that exploit its properties. As stated by Miller and Downing, evolution can produce complex systems "because it can make use of the full, *unmodellable* richness of the physical world"[1]. Giving evolution unrestricted access to the actual CAP allows it to discover and exploit unknown physical properties without being restricted by an imperfect simulator. If it were restricted to a *simulated* CAP, it would ultimately be limited by our own knowledge[1].

# Chapter 3

# Previous work

This chapter examines a few key research advances that have lead to the research of NASCENCE and this thesis. We will begin by reviewing a particularly early contribution by Gordon Pask of the second-generation British cybernetic community in section 3.1. Following, we examine a well known study of Intrinsic Hardware Evolution (IHE) on FPGAs by Adrian Thompson in section 3.2. As a response to Thompson's research, Layzell's development of a general "test bed" for IHE is presented in section 3.3. Building on Layzell's concepts, Harding and Miller's proof of principle of the newly coined term of Evolution in Materio (EiM) is shown in section 3.4. Finally, in section 3.5, we will introduce NASCENCE's hardware and software platform for EiM, Mecobo, which was a central tool of this thesis.

## 3.1 Gordon Pask's electrochemical devices

A key difference between conventional and unconventional computers is that the former are completely specified by a designer, while the latter are to a high degree responsible for evolving their own design through their own creative processes. Because this latter approach does not require a designer to have prerequisite knowledge of how a device should tackle whatever problem it is given, it should ultimately enable the device to out-perform its designer. One of the first to demonstrate this possibility was Gordon Pask who developed several devices capable of evolving their own sensors in the late 1950's[3].

Traditionally, self-organizing devices would learn and adapt according to how the designer makes them see the world. For instance could a device improve its ability to perform its function by learning to increasingly interpret a sound input more accurately. This approach, however, relies on a static *relevance criteria* anticipated and supplied by the designer, in this case the device's microphone. According to Pask, "there is no possible way in which a control mechanism built of elements with well specified functions to perform, can acquire special sensitivity to an input not originally specified as relevant"[3][1]. These insights led to Pask's development

[1] Cariani[3] cites this quote from *The growth process inside the cybernetic machine* by Gordon

of self-organizing devices capable of evolving their own relevance criteria.

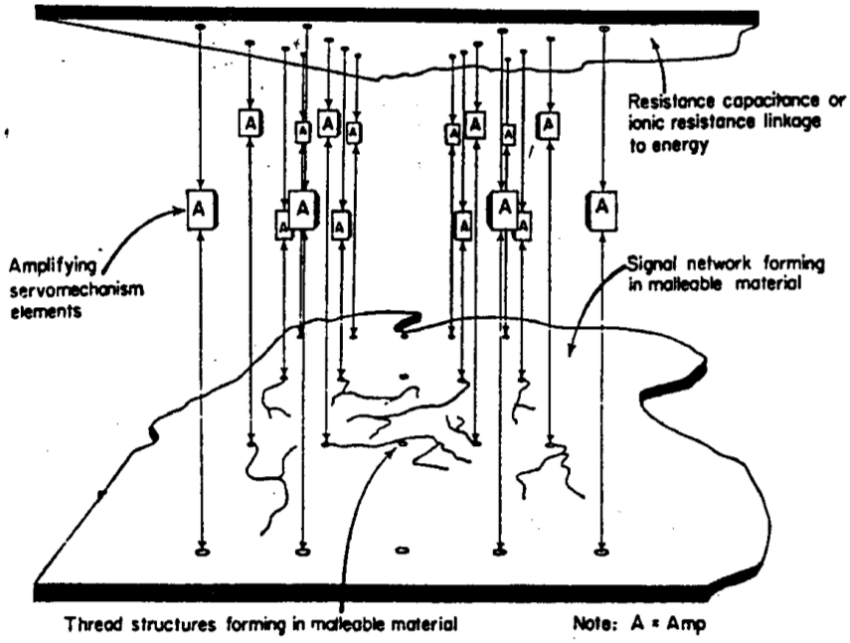



Figure 3.1: Pask's electrochemical device [3]

Pask's first approach to this was finding some computational medium rich in structural possibilities with the ability to alter itself adaptively. He ended up using variations of metals plated out in various solutions, such as acidic aqueous metal-salt. The idea was that the metals could self assemble into wires when probed with various electrical signals. Different set of signals would cause different connection of wires, which leads to different functionality. By controlling the signals through probability-controlled variable resistors, the material could be trained. An illustration of this setup is shown in figure 3.1[2], where the self assembled wires can be seen at the bottom plate.

In his electrochemical device, Pask managed to evolve an ear capable of crude tone discrimination. The device mostly served as a proof-of-concept, and demonstrated that it was indeed possible for a system to evolve its own relevance criteria. A neural network, for instance, could use something like this to independently evolve its own sensors, freeing the designer of the burden of pre-determining all relevant sensors. Like natural evolution, this approach of open-ended hardware design has the possibility to exploit unknown physics and chemistry of the real world[1].

---

Pask, 1958.

[2]Cariani[3] borrowed this figure from *The Natural History of Networks* by Gordon Pask, 1960.

## 3.2 Thompson's evolvable FPGA

In 1996, Adrian Thompson performed "a case-study in intrinsic hardware evolution" and successfully demonstrated that GAs can exploit subtle *intrinsic* characteristics beyond the abstract model of an electrical component[4]. The component in question was a Field Programmable Gate Array (FPGA), a device containing a matrix of logic blocks that can be programmed electronically by a computer. By programming it, the blocks can be connected in arbitrary ways which ultimately allows a programmer to, within some physical limitations, create arbitrary circuits.

Instead of manual programming, Thompson's experiment comprised of using a GA to *evolve* a circuit capable of tone discrimination on the FPGA. The fitness test was conducted on an arrangement as in figure 3.2 and its procedure went as follows: After thoroughly reseting the FPGA, the fitness function (running on the desktop PC) would use the individual's 1800 bits genotype to configure the FPGA. Then, it would command a tone generator to generate a series of 500ms wave bursts at 1kHz and 10kHz in a random sequence into the circuit's input pin on the FPGA. Simultaneously, it would use an analogue integrator to inspect its output pin and compare it with the input waves. The fitness would be evaluated by the circuits' ability to distinguish the two input frequencies. They would have to do this by outputting 0V when exposed to one wave, and 5V when exposed to the other. There was no requirement of which input wave should be marked by which output voltage, so as not restricting the evolutionary process more than necessary. The fitness function was designed to allow small incremental improvements in the evolutionary pathway, and so it would reward partial successes with correspondingly high fitness scores.

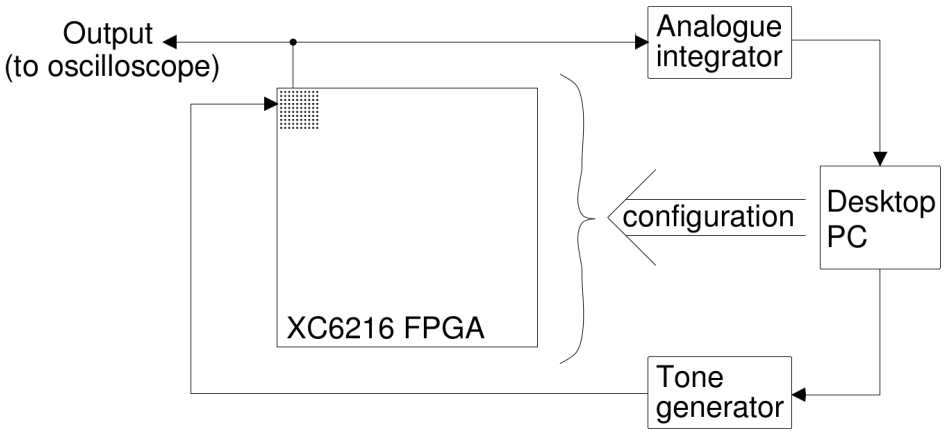



Figure 3.2: Thompson's experimental arrangement[4]

The circuits would have no access to any type of clock or other external components, and would have to fit onto a maximum of 100 of the FPGA's cells, shown

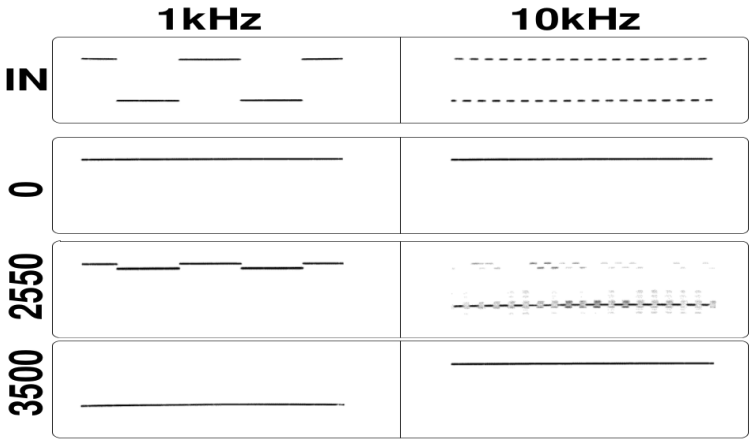



Figure 3.3: GA progress of Thompson's FPGA experiment[4]

in the upper left corner in figure 3.2. The propagation time for a signal to enter and exit one of these cells is only a few nanoseconds, which differs from the input frequencies by five orders of magnitude. As this was a task many people thought would be impossible, evolution was required to come up with some very creative circuits. As it turned out, evolution *did* manage to solve the task: After about 4100 generations, the tone discriminator was visually perfect upon oscilloscope inspection. Figure 3.3 shows a summary of the GA's progress. The top row shows a sample of each input, 1kHz and 10kHz. The subsequent rows shows how the output of the best individual of a small selection of generations steadily improves. At generation 0, it simply outputs 5V regardless of input. Results gradually improves (not shown), and at generation 2550 a partially successful behaviour has emerged: 1 kHz input gives a fairly stable high voltage output and 10 kHz input gives a mostly low voltage output. The behaviour is improved towards perfection at generation 3500.

Aside from the applicability of such a circuit there was one especially interesting outcome of this: It was not possible to impose a digital model upon the evolved circuits. Thompson argues, for instance, that the waveforms for 10kHz input at generation 2550 would seem utterly absurd to a digital designer. He goes on saying "Even though this is a digital FPGA, and we are evolving a recurrent network of logic gates, the gates are not used to 'do' logic." Evolution simply optimizes the circuits for fitness. It is completely ignorant of the fact that the transistors, being arranged into logic gates, are intended to represent a digital model. The GA is thus capable of exploiting the transistors beyond their digital intent.

Electronics in general must conform to their abstract model in order to be accepted as a correct product. However, electronics are never perfect and even though the model holds, there are always irregularities on the microscale. An example circuit produced in the experiment "was shown to be using subtle interactions between adjacent components on the silicon". When this same circuit was used to configure

another physical region of the FPGA, having the exact same digital model, its fitness deteriorated by about 7%. Some circuits only deteriorated by about 0.1%, and most quickly recovered when the GA was continued, but the fact that there was a difference at all proves that the circuits exploited subtle physical properties of the transistors, beyond their digital model and intent.

## 3.3 Layzell's "test bed" for intrinsic hardware evolution

After Thompson's successful demonstration of Intrinsic Hardware Evolution (IHE), Layzell argued that FPGAs were not an ideal medium[5]. For example, the FPGA circuits are digital and even though it is possible to exploit the analogue aspects of them, it may cause the resulting circuits to "suffer from a number of undesirable characteristics such as dependence on temperature and lack of portability", which was the case with Thompson's circuits[4]. Also, since it is not possible to observe the individual circuit elements of the FPGA, evolved circuits can be extremely difficult to analyse. To address these and other issues, Layzell conceived the concept of the Evolvable Motherboard (EM).

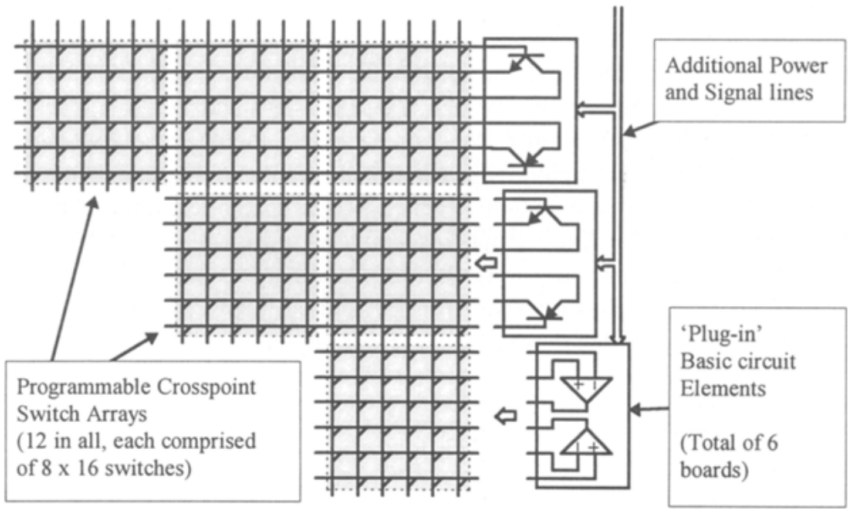
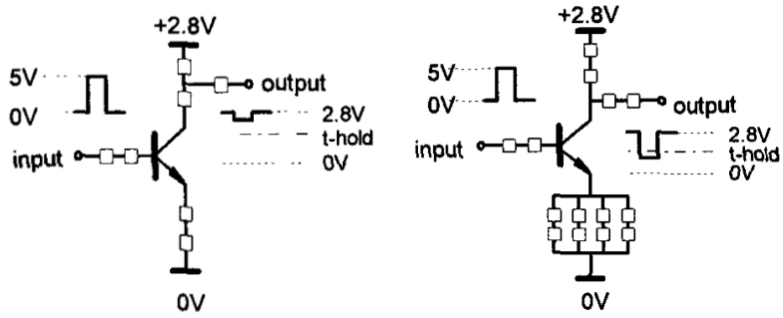



Figure 3.4: A simplified representation of the Evolvable Motherboard[5]

Layzell described his EM as a test bed for IHE. Through some initial experiments, he demonstrated that the EM can be a valuable tool to "investigate many important issues arising in current IHE research, including analysis; fault-tolerance; genotype encoding; portability; basic elements, and evolved topologies." The proposed EM, shown in figure 3.4, consists of a diagonal matrix of analogue switches with 6 sockets for plug-in daughterboards containing the basic circuit elements of the CAP. In the figure, these basic elements are transistors and operational am-




(a) Original hand-seeded (b) Fittest individual

Figure 3.5: Circuit diagrams for first experiment[5]

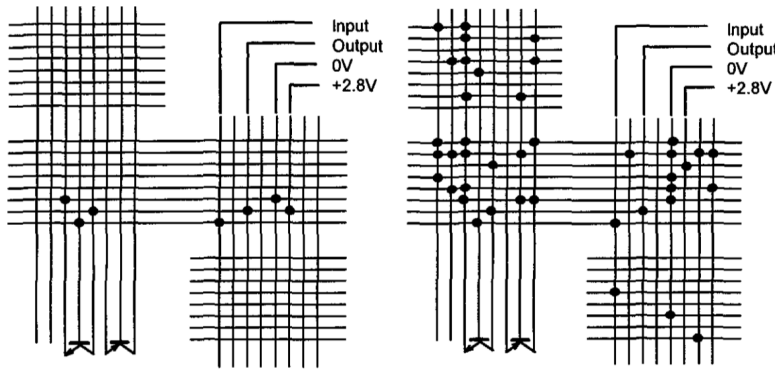
plifiers, but any type of component can be used. The switch matrix is designed to support any combination of interconnection between the basic elements in the daughterboards, and it provides a total of $10^{420}$ possible circuits.

The practical advantages of the EM are significant: It can be subdivided by software so that a configuration can simultaneously be mapped to multiple sets of components, allowing a practical way to incorporate portability into fitness; it contains additional connectors allowing several EMs to be daisy-chained together; it plugs directly into a host PC's ISA bus, giving a configuration delay of less than 1ms; and it allows for easier circuit analysis than on an FPGA, though not completely without difficulty.

Layzell demonstrated the EM with three experiments all involving the evolution of a NOT gate. In the experiments, transistors were used as the evolutionary building blocks. They all used a GA with single-point crossover, rank-based selection, and elitism. The fitness function tested the circuits by applying 50 logical lows and 50 logical highs in a random sequence into the circuit and recording the output voltage. The fitness was calculated, as shown in equation 3.1, by taking the sum of all output voltages $v_t$ obtained when a low voltage was applied as input (and a high voltage was expected as output), i.e. $t \in S_L$, and then subtract from this the sum of output voltages where a low was expected, i.e. $t \in S_H$. This would reward every test by how close the output voltage came to the desired response.

$$f = \frac{1}{5}\left\{\left(\sum_{t \in S_L} v_t\right) - \left(\sum_{t \in S_H} v_t\right)\right\} \tag{3.1}$$

In the first experiment, the initial population was seeded with a hand-designed circuit to see how evolution could improve it. The hand-designed circuit, shown in figure 3.5a, "conforms to the NOT function in that its output corresponding to a '0' input is of slightly higher voltage than that corresponding to a '1' input, however this difference is too small to be of any practical use."




(a) Original hand-seeded (b) Fittest individual

Figure 3.6: Circuits instantiated for the first experiment[5]

During the course of 470 generations, the best fitness increased from $\sim$ 11% to $\sim$ 34%. The circuit of the best individual, shown in figure 3.5b, is clearly a modification of the initial hand-designed circuit. Evolution improved the original circuit by increasing the resistance between power and the transistor's collector, and decreasing the resistance between the transistors' emitter and ground. The switch matrices for the original hand seeded and the best individual are shown in figures 3.6a and 3.6b respectively. We see that closed switches, represented by black dots, are mostly *added* to the original design, and only one is removed from the original. These switches behaves like low-value resistors, so evolution has mostly added resistors, but has nevertheless kept its ability to both increase and decrease resistance by exploiting a fundamental property of resistors: To increase the resistance, resistors are added in series, and to decrease it they are added in parallel. This way, it was able to tweak the resistances of the original circuit in a way that improves its fitness.

The second and third experiment evolved the NOT gate from scratch, and reached a much higher fitness. Rather than to draw conclusions of the phenomena observed and their implications, Layzell's experiments are intended to illustrate the capabilities of the EM as a research tool. Layzell argues that, because of the tendency of evolution to evolve difficult-to-analyze circuits, relatively small and simple circuits such as those produced in the EM "continues to play a major role in the development of IHE". Using the EM, many important issues arising in IHE research can be investigated.

## 3.4 Harding and Miller's liquid crystal tone discriminator

In 2002, Miller and Downing argued that "artificial evolution of hardware might work much better if we can create ways of exploiting the rich resident physics of materials or complex systems"[1]. In the same paper, they coined the term Evolution in Materio (EiM), and described a form of CAP known as a Field Programmable Matter Array (FPMA). An FPMA is a device where a set of wires is connected to a rich physical substrate. The idea is that voltages applied to the substrate may induce physical changes that interact in unexpected ways with other distant voltage induced configurations. There were many possibilities for such a substrate, among them Liquid Crystal (LC), which was described as an obvious candidate. LC can exist in a mesomorphic state, i.e. a state that is simultaneously liquid and solid. In a solid, there is long range order for both orientation and position of molecules and in a liquid there is for neither. The molecules of LC, when in its mesomorphic state, has long-range oriental order, but no long-range positional order.

In 2004, Harding and Miller followed the suggestions of Miller and Downing and showed that "liquid crystal can be used as a medium for intrinsic hardware evolution", which "demonstrates proof of principle of in materio evolution"[6]. Using Layzell's concept of EM, they developed an FPMA with LC, which they successfully used to evolve interesting and complex behaviour. Their device, the Liquid Crystal Evolvable Motherboard (LCEM), consisted of a monochromatic matrix Liquid Crystal Display (LCD) with a 180x120 pixel resolution interfaced by 64 wires which could be controlled by four 8x16 analog switch arrays. The schematic of the LCEM is shown in figure 3.7.

Figure 3.8 shows the different layers of a typical LCD. The LC is represented by the middle layer (c). This layer is sandwiched in between layers (b) and (d), which contains electric connections. The outer layers, (a) and (e), are polarising filters, one horizontal and one vertical. In a typical LCD, a driver circuit would apply constant voltages, high or low, to electric connections in order to turn a pixel, i.e. a small area of the LC, on or off. In Harding and Miller's experiments, this driver circuit was replaced by their LCEM. It would not only have the possibility to apply arbitrary voltages to the LC, but also record voltages from it. This way, the LC could function like an FPMA.

In their experiments, they sought to evolve a non-linear function, a transistor, and a tone discriminator. As with Layzell's EM, the genotype contained the connectivity of the switch matrix. Additionally, a second part of the genotype contained the configuration voltages, -10V to +10V, to be applied to the LCD independently of input signals. This genotype allows evolution to control which LCD connector the input, output and configuration signals should connect to and the voltage of each configuration signal. Additionally, it can control which LCD connectors should be grounded.

In the first experiment, a non-linear function was evolved using a GA. The fitness was measured by applying a voltage steadily increasing from 0V to 3V over the course of 2ms into the input pin of the LC. During these 2ms, three samples

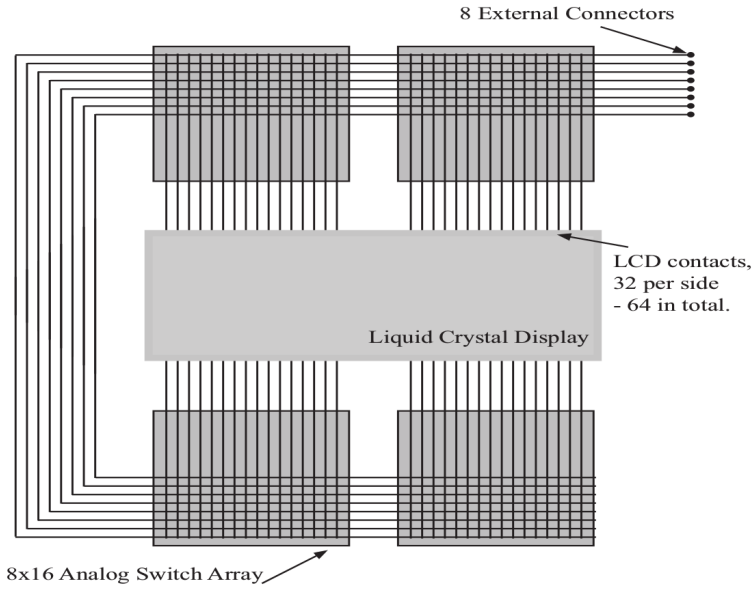



Figure 3.7: Schematic of LCEM[6]

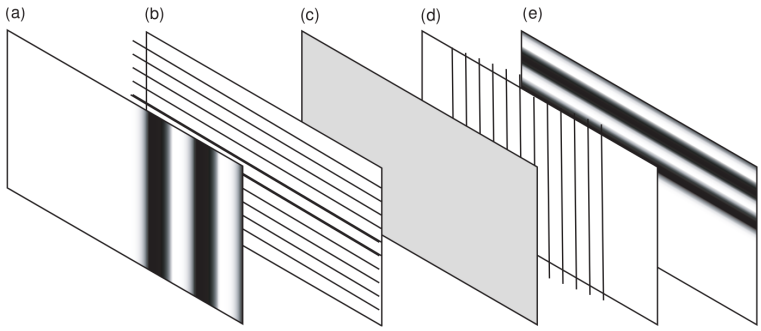



Figure 3.8: Layers in an LCD[6]

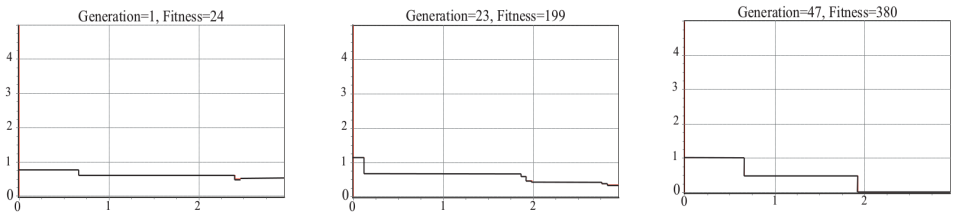



Figure 3.9: Evolution of a non-linear response[6]

would be made from the output pin. In essence, the fitness was defined as the sum of the differences between the output sample and corresponding input voltage. This means that a measured decreasing voltage would score high, while an increasing voltage would score low. Parts of the evolutionary run can be seen in figure 3.9, where the Y axis shows the measured voltage and the X axis the voltage applied into the LC[3]. The authors did not manage to reproduce similar results by using random search. They also attempted to repeat the experiment without the LC connected. This failed, which is an indication that the LC was responsible for the non-linear functionality.

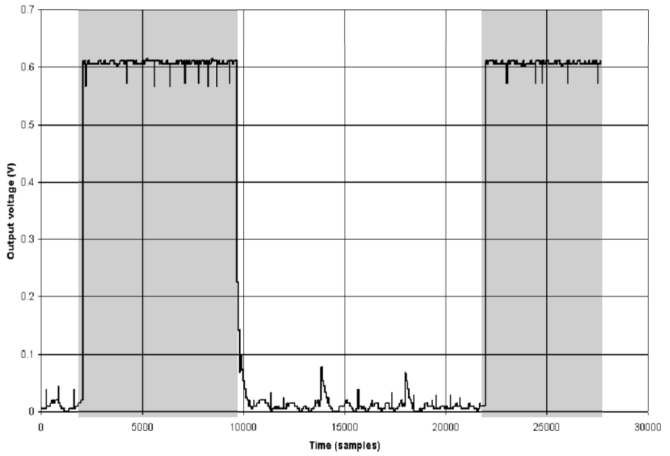



Figure 3.10: Tone discriminator response[6]

Another experiment, loosely based on Thompson's FPGA experiment[4] discussed in section 3.2, comprised of evolving a tone discriminator in the LCEM. The input frequencies was arbitrarily chosen as 100Hz and 5kHz in a square wave. The desired functionality was an output voltage $< 0.1$V for the 100Hz input, and $> 0.1$V for the 5kHz input. Though not all attempts were successful, they managed to evolve a response as in figure 3.10. In the figure, the dark areas indicate

[3]As the input voltage steadily increases over time, the X axis can also be viewed as time.

5kHz input, and light areas 100Hz. Even though the output signal is not stable, there is a clear distinction between responses of the different input frequencies. The authors are "of the opinion that the behaviour stems from the capacitative effects originating inside the LCD, and that the system is acting as a form of RC network." It is unlikely, they argue, that the crosspoint switches are involved as they are designed for high frequency audio/video signals: Both feed-through capacitance and switch I/O capacitance are too small to have any filtering effect on the low frequencies used by the GA. They also observed on some of the partial solutions that the response to a change in tone takes some time. This means that a capacitive effect is likely to be the cause of the behaviour.

## 3.5 NASCENCE's EiM platform

The Mecobo platform is a hardware and software platform for EiM developed by NASCENCE[7, 11]. This platform contains an FPMA, and EiM is performed by exploiting the computational potential of unknown components in much the same manner as Harding and Miller's LCEM discussed above. This means that electrical signals, e.g. constant voltages, square waves, and sinus waves, are sent through various materials. In this scheme, the multiple signals are sent through simultaneously, some of which represents the program, i.e. the *configuration*, and others representing the information to be computed. The materials in question are currently different concentrations of carbon nanotubes randomly deposited on a glass slide. This is a solid material that cannot be physically re-organized during optimization. The advantage of a solid material is stability and determinism: The only possible state will be of relatively volatile charges in the material that will affect time-dependent signals, but not the subsequent individual probings[22].

Lykkebø et. al. introduced Mecobo by using it to demonstrate the computational potentials of carbon nanotubes[7]. In the first experiment, they performed an exhaustive sweep of all possible configurations, limited to digital time-independent signals. This way they could map out all possible two-input logic gates that could be implemented in the material. For a 12 pin material interface, one pin was used as logic gate output, two as logic gate input, and the remaining 9 for the configuration vector. All such input- and configuration vector to output mappings would then be organized by common configuration vectors into *gate output sums*. An example of a gate output sum for one configuration vector is shown in table 3.1. The gate output sum is the decimal representation of the concatenation of the binary digits in the output column. The gate output sum in the example table is 6 because $6_{10} = 0110_2$. Comparing inputs with outputs we can see that this gate is an exclusive or (XOR). In figure 3.11 all the possible gate output sums calculated from the exhaustive sweep using every pin configuration are plotted onto a graph. The X axis represents the configuration vectors and the Y axis the gate output sums, both in decimal. By this method, they were able to visualize the functionality of the material, within the restrictions of two-input logic functions and time-independent signals. The results shows that the XOR gate (6) was rare, though present in at least one configuration. Other gates, such as NAND and NOR

| Input | Config | Output |
|---|---|---|
| 0,0 | 1,0,1,1,0,1,0,1,1 | 0 |
| 0,1 | 1,0,1,1,0,1,0,1,1 | 1 |
| 1,0 | 1,0,1,1,0,1,0,1,1 | 1 |
| 1,1 | 1,0,1,1,0,1,0,1,1 | 0 |

Table 3.1: Gate sum mapping of XOR[7]

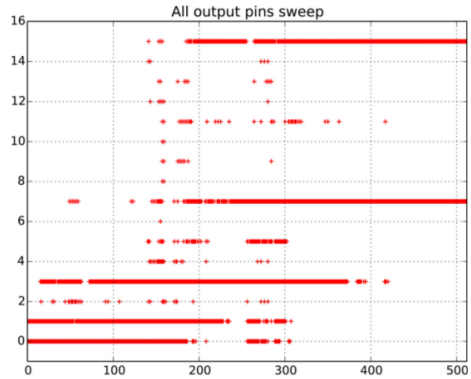
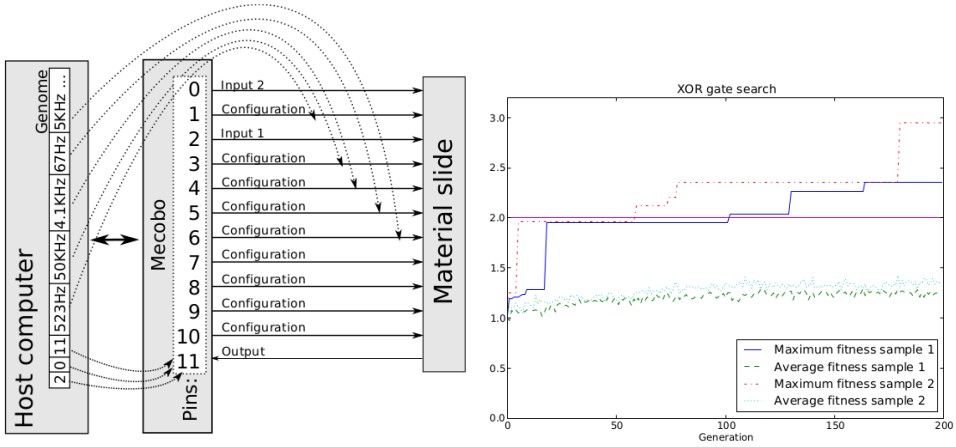



Figure 3.11: All possible gate output sums [7]

(7 and 1 respectively), where abundantly available. This demonstrates that some gates would be easier to find than others, but every gate was indeed possible to find and the sample was thus "capable of solving problems beyond simple threshold functions."

A second experiment demonstrated the presence of evolvability. Figure 3.12a, showing the genome representation, illustrates the parameters placed under evolutionary control. In the genome, the first two genes assigns input pins, in this example pin 2 and 0. Similarly, the next gene assigns the output pin to pin 11. The remaining pins are assumed to be configuration pins, and the remainder of the genome holds the configuration represented as, for each configuration pin, a frequency between 400Hz and 25MHz. For instance, the first configuration pin, i.e. pin 1, will apply a 523Hz square wave to the material. The GA was set up to evolve a configuration capable of using the material as a 2-input XOR logic gate that, unlike the exhaustive sweep, had a requirement of *stability*. The evolutionary progress for two material samples is shown in figure 3.12b. The graph illustrates that, over the course of the generations, the maximum fitness of both samples rises in bursts every now and then. The average fitness seems to have a more gradual, though slower, ascent. Though maximum fitness was not reached in sample 1, this evolutionary progress demonstrates that for both of the attempted material samples, evolvability was present.




(a) Representation mapping for GA (b) GA progress

Figure 3.12: Experimental setup for genetic search[7]

# Chapter 4

# Methodology

This chapter describes the technology that was developed and its utilization as experimental tools. The discussion centers around a material interface that we developed to electrically isolate the material from the surrounding Mecobo hardware. We begin in section 4.1 by introducing the purpose and requirements for such an interface. Following, in section 4.2 we will take a closer look on this surrounding hardware, i.e. the Mecobo platform, and see how it works.

A high-level introduction to the developed material interface is then presented in section 4.3 where we discuss its circuit design and how it works. Further, the process and result of implementing this circuit in hardware is documented in section 4.4, and the steps taken to integrate it into Mecobo in section 4.5.

Having the design and implementation of the interface covered, we will then, in section 4.6, document some of the problems arisen from the use of opto-isolators and how they affected the design process. Finally, in section 4.7, the utilization and determinism of the interface used as an experimental tool is discussed.

## 4.1 Requirements of interface

The purpose of the material interface is ultimately to deny any GA access to exploit any physical properties in the surrounding electronics of the material. The surrounding electronics, in this particular case, is the Mecobo platform. It is a relatively complex piece of hardware, containing both an FPGA and a microcontroller. To avoid electrical side-effects of Mecobo to affect the material, the material must not be electrically connected to Mecobo. This could be achieved by designing and implementing a material interface that electrically isolates the material from Mecobo while allowing EiM to function as normal. The chief component of the implementation of this interface would be the opto-isolator. It is an off-the-shelf component that transfers an electrical signal from one circuit to another by the use of light.

As this is the first effort to electrically isolate the material in EiM, the task is simplified to only involve digital signals.

### 4.1.1 Requirement 1

The first and foremost requirement of the material interface is directly related to the purpose of this thesis: **The interface should electrically isolate Mecobo from the computational material**. As mentioned in chapter 1, when the material and its "surrounding hardware", e.g. Mecobo, are not electrically isolated, the GA cannot distinguish whether the behaviour it senses is a result of the material alone or the material in combination with the surrounding hardware. As stated by Lykkebø and Tufte, a challenge to EiM is "the definition of the border between the apparatus applying configuration signals, and the material itself"[23]. By electrically isolating the material from Mecobo, we can essentially draw such a defined border.

### 4.1.2 Requirement 2

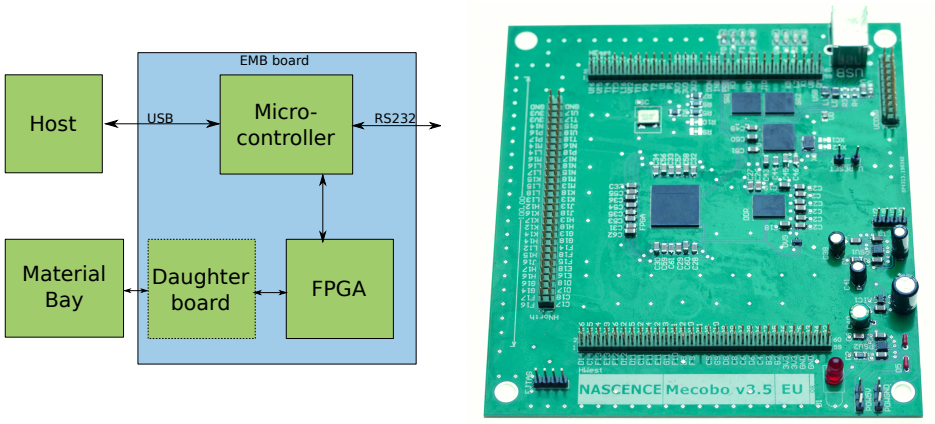
**The interface should not significantly disrupt the possibility to perform EiM on the material**. For instance, we should seek to maximize the interface's ability to transfer data and minimize its effect on the computational properties of the material. The interface must also remain fully controllable by the Mecobo system and not require manual intervention. The reason for this is to avoid placing restrictions on the GA: A crucial aspect of EiM is to explore intrinsic unknown properties in a component, and to transfer control of said component from the GA to the designer would be contradictory. GA control of pin selection, i.e. which pins are input and output, is particularly relevant. The favoured approach in the design of Mecobo was to put pin selection under evolutionary control[7, p. 271].

## 4.2 The Mecobo platform

Figure 4.1 shows a block diagram and a photograph of the Mecobo platform.[1] The *Host* component of figure 4.1a represents a PC which is connected to the Mecobo board by USB. It can command Mecobo to apply signals to, or record output from, the material. Typically this PC either runs some GA or it functions as a server exposing Mecobo to other PCs over the internet. The actual board, shown in figure 4.1b and enclosed in the blue square labeled *EMB board* in figure 4.1a, has a Field Programmable Gate Array (FPGA) and a microcontroller as its main components. The microcontroller functions as a USB interface and receives the commands from the host PC. The microcontroller communicates these commands to the FPGA, which in turn applies them to the material. The FPGA functions as an interface to the material and is able to route any signal to or from any pin; there are no restrictions to whether a material pin is input or output.

Also included in figure 4.1a is the optional *Daughter board* component. A daughter board can be placed in between the FPGA and the material to add further functionality to the platform without having to redesign everything. In this

[1] The photograph by NASCENCE.




(a) Overview of Mecobo[11, p. 20] (b) Photograph of Mecobo

Figure 4.1: Mecobo

thesis, the aforementioned material interface is implemented as such a daughterboard. A photograph of Mecobo with the final material interface implementation, named optowall, is shown in figure 4.2. In this photograph, the optowall daughterboard is plugged directly into Mecobo's 60 pin header north. The material slide, representing the *Material Bay* component of figure 4.1a, is seen at the far right, as a part of optowall.

## 4.3 Circuit design of material interface

The circuit of a single material pin interface is illustrated in figure 4.3. This circuit is repeated for all connected pins to the material. A key aspect of requirement 2, i.e. not disrupting the possibility to perform EiM, is that every pin should be considered a duplex pin, i.e. one that can be used as both input and output. For this reason, all pins contains components both for applying input and reading output.

Because the purpose of the interface is to separate the material from the Mecobo circuit, it consists of two separated circuits. The left hand side is the part which is connected to the Mecobo circuit. The right side, separated by the dashed line, is the material circuit. Data transfer between the Mecobo- and material-side circuits is made possible by three opto-isolators per material pin. The following sections describe the workings of the circuit by means of two use-cases: Reading output from the material, and applying input into the material.

### 4.3.1 Reading output from material

At the bottom left of figure 4.3 the *opto-out* component, comprised of one opto-isolator, connects material output to mecobo. A logically high voltage from the

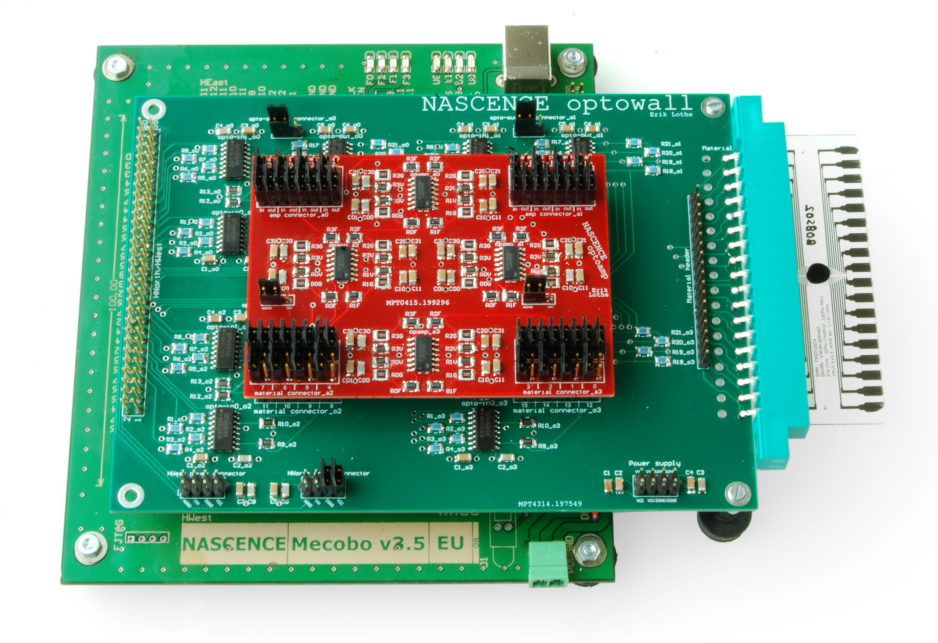



Figure 4.2: Mecobo with optowall as daughterboard

material lights the diode of the opto-isolator which is detected by the phototransistor on the Mecobo side. When the phototransistor is triggered by the light, it allows current to flow from VD+ to VD-. This causes the voltage at the *out* signal to fall low. Inversely, *out* will be high when there is no light from the diode, because the transistor will block the current and all the current from VD+ will flow to the *out* signal instead. The *out* signal is routed back to mecobo, and can be interpreted as an inverse ouput from the material.

For reasons discussed shortly, the material output signals needs amplification to aid their translation through opto-out. This amplification is performed by the *amplifier* component in the upper right corner of figure 4.3. It is comprised of an operational amplifier with a feedback loop and a reference voltage controlled by two resistors. To control the reference voltage, a static voltage divisor was chosen over a potentiometer for two reasons. First, it is much easier to keep multiple amplifiers at the same setting without potentiometers, and second, the manual control gained of the signals by potentiometers would not be accessible to the GA or any other software. The material pin, which in the output scenario represents the signal source, is connected into the amplifier which, on the other side, drives the opto-out component.

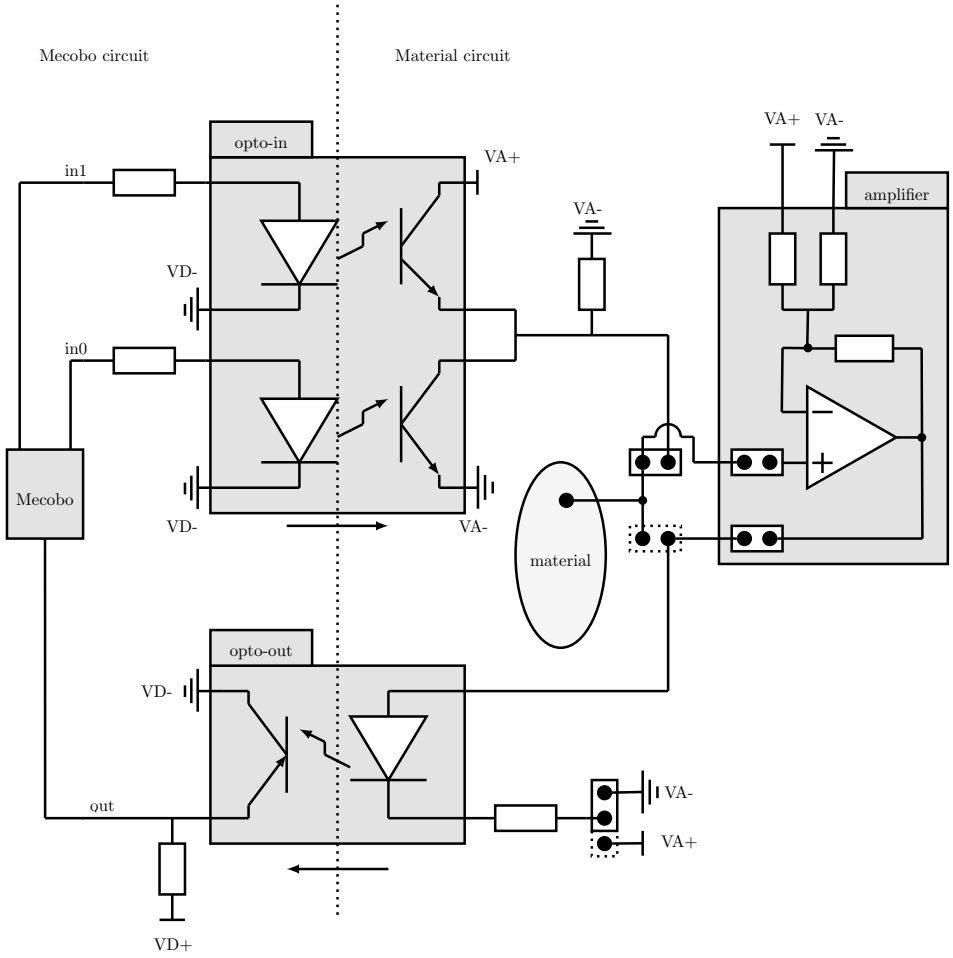



Figure 4.3: Material interface circuit

| Mecobo input array | Material pin value |
|---|---|
| 01 | 0 |
| 10 | 1 |
| 00 | Z |

Table 4.1: Relationship between input and output of the opto-in opto-isolator

### 4.3.2 Applying input to material

The two top opto-isolators, labeled *opto-in*, connects a signal to the material from Mecobo as input. There are two opto-isolators to allow a defined logical low. This is important because the material is essentially analogue; small differences in voltage can possibly change the behavior. This is as opposed to the digital Mecobo side, where as long as the voltage is measured to less than 0.8V, it is interpreted as logical low by the LVCMOS33 standard used on Mecobo[7, p. 272]. Because every material pin needs to be represented by two values at the input side of the opto-isolator, Mecobo needs to represent each input signal to the material as a two bit array where the rightmost bit represents the value of $in_0$ and the leftmost $in_1$. The relationship between input arrays from Mecobo and the values that are applied to the material pin is then as in table 4.1, where the least significant bit represents in0.

Let us take the input string of 01 as an example. This input causes the upper phototransistor to close, meaning that current can not flow from VA+. Additionally, the lower phototransistor opens, meaning that any potential on the material pin is drawn to ground (VA-), effectively turning the pin into a sink. Inversely, an input of 10 allows current from VA+ to enter but not to exit through VA- and is thus forced to enter the material. Additionally a state of high impedance (Z) can be achieved by inputing 00 which closes both phototransistors, essentially disconnecting the material pin from the input opto-isolators. This latter state is particularly useful when the pin should be used as an output pin.

## 4.4 Hardware realization of material interface

The material interface is realized by two boards, the optowall daughterboard and the optoamp extension board. Plugged together, they consist of 16 instances of the circuit introduced above, making them capable of connecting up to 16 material pins.

The optowall daughterboard is the main tool. It is designed as a daughterboard for Mecobo and can be plugged directly into one of its I/O connectors. It is also the board in which the material is connected. The optoamp extension board is a later addition made up chiefly of operational amplifiers that can amplify output signals. In addition to the designing of these boards, we have also hand soldered and validated both boards and, to some degree, Mecobo. Software was also developed to allow a quick, easy, and automated utilization of the hardware tools. As mostly irrelevant to the discussion, they are only discussed in appendix C. The schematics

of both PCBs are included in appendix A.

### 4.4.1 Optowall

From the perspective of evolution, the optowall daugther board is essentially a wall denying access to anything else than the material for exploitation. It consists of a Printed Circuit Board (PCB) with 12 quad channel opto-isolators, a connector for an electrode glass slide containing the computational material, connectors to external power supplies, various jumper connectors for increased flexibility, and a large bus connector that is compatible with two of Mecobo's three I/O headers.

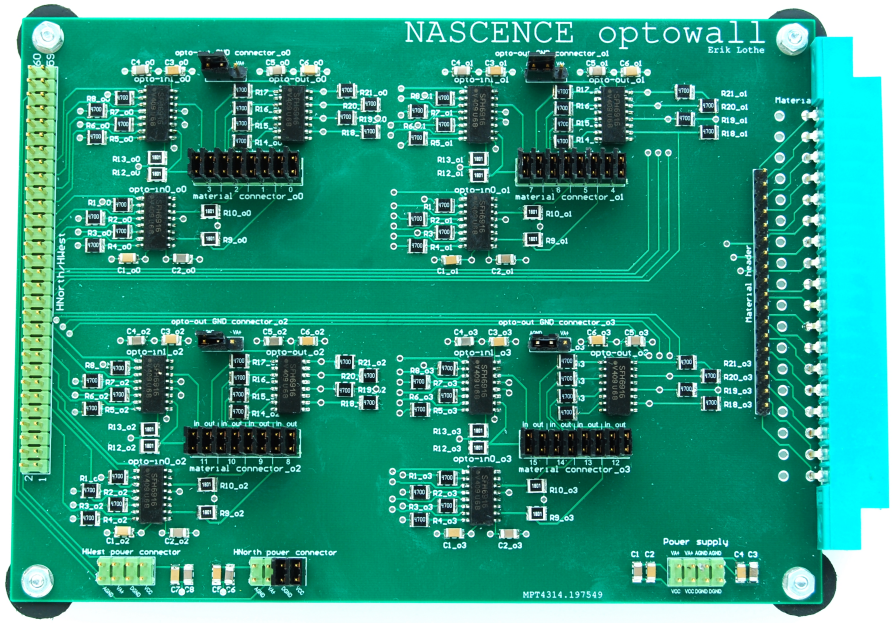



Figure 4.4: Optowall

A photograph of the board is shown in figure 4.4. It is organized as follows: The main I/O bus connector is placed at the far left side. Most of the pins are used for data lines, except those that are either power pins or not used at either Mecobo headers Header North or Header West. All power pins are routed to two optional power connectors which are discussed below. The central part of the board consists of four groups of opto-isolator circuits. Each group has two quad channel opto-isolators for *opto-in* and one for *opto-out*. Each group connects to four material pins.

**Power considerations**

Because optowall is essentially two separated circuits, both circuits needs their own power supply and the PCB is designed to support this in a flexible way. There are four power nets in total which all have their own power plane inside the PCB: Digital high and ground for the Mecobo-side circuit, and analogue high and ground for the circuit side. Power can be supplied to the nets by connecting external power supply cables to the power supply header on the bottom right corner of the PCB, which is shown in figure 4.5. This header has two pins for each power net so that power nets also can be connected together through jumpers. For instance could both ground nets be connected so that both circuits would share the same ground.


Figure 4.5: Power header


Figure 4.6: Mecobo power connectors

Since the Mecobo-side circuit transfers data to and from Mecobo, the digital power should in most cases come directly from Mecobo and not an external power supply. The two headers HWest_power_connector and HNorth_power_connector in the lower left corner of the PCB can draw power from Mecobo header west and north respectively. These headers, shown in figure 4.6, allows Mecobo's power and ground to be connected to either of optowall's power and/or ground planes, AGND, VA+, DGND, and VCC, by connecting jumpers on the corresponding pins.

### 4.4.2 Configuration flexibility

On the material side, both the input and output opto-couplers are connected to the material through a header connector labeled "material connector". This was added as a safety feature in case there would be any problems attaching a material pin to both input and output at the same time. If so, we could simply remove the jumper that connects them. In figure 4.3, these jumpers are the two square boxes connecting to the material. Additionally, as was later proven useful, extra components such as amplifiers could be connected to the material interface at these connectors. The ground signal on the diode on the opto-out opto-isolator can also optionally be connected by jumpers to VA+ instead of VA-. This would allow the

diode to be considered a part of the computational medium, which might be an interesting configuration for the in materio evolution.

### 4.4.3 Optoamp

To facilitate amplification of all material pins in a stable and reliable manner, we designed another board, the optoamp extension board. It mimics the design of optowall by using four quad channel operational amplifiers to amplify all 16 material pins. It can be stacked onto optowall through its material connectors, as shown in figure 4.7. It gets its data through half of the pins on the material connector representing the connection between material and opto-in. These headers are illustrated in figure 4.3 as the solid-line square box connecting to the material. To also allow input signals from opto-in to enter the material, these pins should be connected by jumpers as usual, and optoamp facilitates this by extending the header so that the jumpers can instead be placed on optoamp with the same effect. The remaining pins, which are represented by the stippled lined box of figure 4.3, should not be connected, so that opto-out is only connected to the output of the amplifier, and not the material directly. As the board would need to be placed physically on top of optowall, it would also block two of the opto-out GND connectors located in between the material connectors on optowall, making it difficult to place jumpers on them. Therefore, these connectors were also extended to optoamp. Conveniently, these connectors contain both analog voltage supply and analog ground, and are thus used as power supply for optoamp.

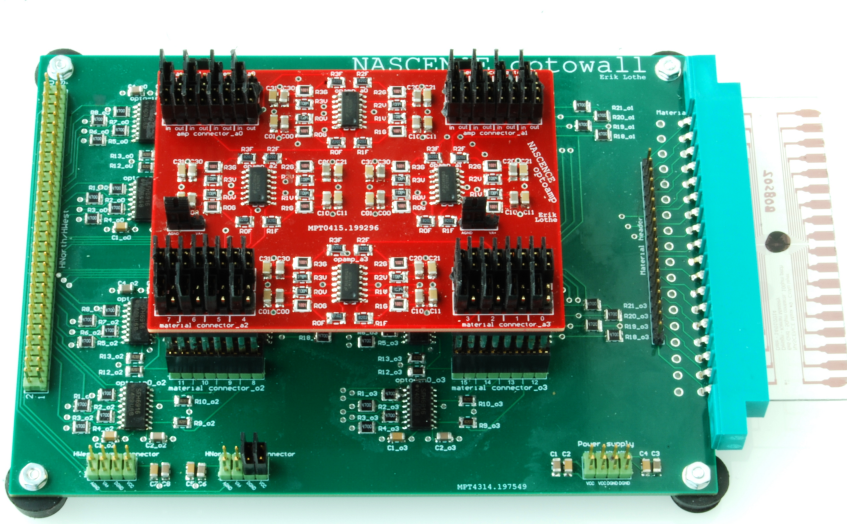


Figure 4.7: Optowall with the optoamp extension board

## 4.5 Integrating the optowall daughterboard with Mecobo

Because of optowall's unusual I/O scheme, as discussed in section 4.3, some sort of interface was required to integrate optowall with Mecobo. This section describes how this integration was achieved.

### 4.5.1 Integrating Mecobo and optowall through software

Initially, it was attempted to interface Mecobo and optowall through software on the client side. Every input value to be sent to the material would first be mapped to a function equal to table 4.1, returning its corresponding two bit value. Second, the material pin to be written to would be mapped to a function returning the corresponding two HNorth input pins on Mecobo that connects to the given material pin. Finally, a command would be sent to Mecobo that applied the mapped values to the mapped pins which would assert the proper value on the proper material pin. This scheme worked fine when applying constant voltages. With waves, however, the lack of accurate timing on Mecobo's scheduler was a critical problem. The source of the problem was located in the preexisting software on Mecobo's microcontroller. It would apply signals to the FPGA by continually picking *items*, each representing a signal to be applied to or read from a pin, from a queue in an infinite loop.

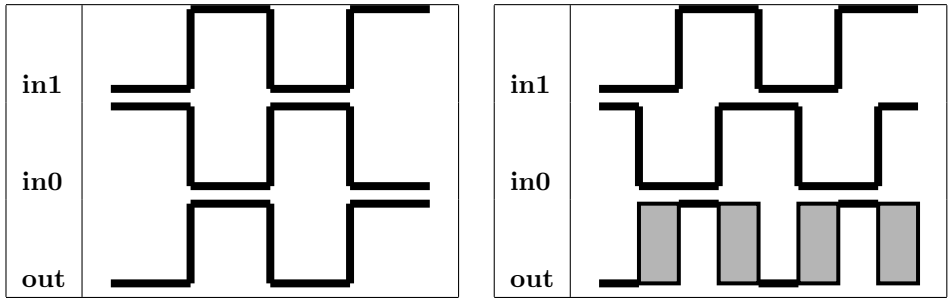
```
for (;;) {
    ...
    struct pinItem * currentItem = &(itemsToApply[itaPos]);
    //Is it time to start the item at the head of the queue?
    if (currentItem->startTime <= timeMs) {
        execute(currentItem);
        ...
    }
    ...
}
```

Listing 4.1: Item execution on Mecobo's microcontroller

As shown in listing 4.1, the loop body would compare the item's start time with the current time and, if the start time was less than or equal, execute the item. Not shown in the listing is a lot of additional work that was performed at every iteration. As the timing of this time comparison depended on the variable amount of work performed in every loop iteration, items would be executed at unpredictable times after their scheduled start time.[2]

Because of the nature of the optowall circuit (discussed in 4.3), applying a wave to the material through optowall would require two opposing square waves so that while one was high the other should always be low. As illustrated by figure 4.8,

[2]NASCENCE has subsequently fixed this problem.




(a) No skew (b) Skew

Figure 4.8: Waves translating through opto-in

to properly represent the output wave, the input waves could never be *both* low or high. Figure 4.8a shows that two perfectly opposing input waves on each of the input opto-isolators, denoted in1 and in0, produces a clean output wave onto the material pin. In the case of a skew on an input wave, as illustrated in figure 4.8b, the output wave will contain several illegal states, marked by the grey boxes. The first of such states, having both input waves momentarily low, will cause the output to go to a state of high impedance, which differs from the desired defined low. When both inputs are high, the output goes into a state that was never intended to be used. This state would cause both phototransistors to open, allowing current to flow almost freely from the voltage source to ground, possibly damaging the phototransistors. As the microcontroller would execute the two opposing waves at unpredictable times, it was impossible to reliably achieve exactly opposing waves in practice. To fulfill this requirement, the FPGA would need to be modified.

### 4.5.2 Modifications to the Mecobo FPGA

The modifications to the FPGA would move the responsibility of interfacing optowall's unusual I/O scheme from the client software past the microcontroller to the FPGA. In addition to circumnavigating the microcontroller's timing issues, having the FPGA perform all pin abstractions would make optowall compatible with all preexisting software.

The changes to the FPGA mainly concerned its `pincontrol` module. Originally, each instance of this module would contain a reference to one physical pin and signals to control whether the pin should function as input or output.[3] When output was enabled, the value on the pin would be driven by a state machine, `pin_output`, which would drive the signals instructed by the microcontroller. The module was changed so that instead of one duplex pin per pin controller, it would have one input and two output pins. The outputs would then be driven according to table 4.1 (page 48): When output was enabled it would drive one pin to the value of the state machine, and the other to its inverse. Otherwise, both pins

[3]In this subsection, "input" and "output" is relative to Mecobo.

would be driven to zero, causing a state of high impedance on the material pin. Additionally, it would invert the input, as this signal is inverted by the opto-out component. Listing 4.2 shows the key modifications in Verilog code. The main clause of the compiler directive `ifdef` shows the added code while the else clause shows the preexisting code. Additional code elsewhere maps the proper pins to the modified pin controllers.

```
‘ifdef WITH_OPTOWALL
    //Drive output pins to 01 (high), 10 (low) or 00 (Z)
    assign pins_out  = (enable_pin_output)?
            ((pin_output)?  2’b01 : 2’b10): 2’b00;
    assign pin_input = ~pin_in;
‘else
    //Drive output pin from pin_output state machine if output
    assign pin = (enable_pin_output) ? pin_output : 1’bZ; //Z or 0
    //else we have input from pin.
    assign pin_input = pin;
‘endif
```

Listing 4.2: FPGA pin controller modifications

With this solution, there was no observable timing issues when viewed at an oscilloscope with 5 ns resolution. As the responsibility of correctly timing the two waves was moved to the FPGA, the timing issues of the microcontroller could be disregarded.

## 4.6 Problems arising from the use of opto-isolators

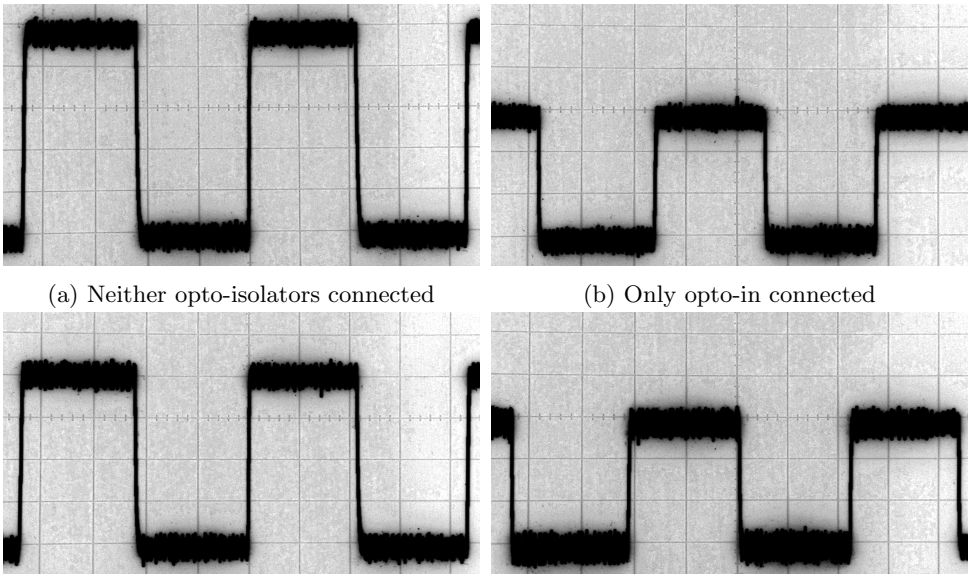
Because the nature of the material as a computational medium is inherently sensitive, applying opto-isolation to its interface introduced some issues. This section addresses these issues and describes how they where counteracted.

### 4.6.1 Voltage drop on material output

Initially, the material interface was implemented without any amplification of the output signals. During the initial testing of the interface, which at the time existed solely in the form of the optowall daughterboard, it was discovered that as more material pins were connected to the opto-isolators, the amplitude of output signals became lower.

To investigate this phenomena, we sent a square wave into pin 0 and measured the output on pin 1 with four different jumper configurations on pin 2, i.e. the closest unused pin. All remaining pins, pin 3 through 15, had all of their opto-isolators, i.e. both opto-in and opto-out, disconnected at all times. The output waves in the four different configurations are shown in figure 4.9.

The first configuration, whose output is shown in figure 4.9a, functions as a negative control, i.e. a configuration in which no phenomena is expected. In this



(a) Neither opto-isolators connected

(b) Only opto-in connected

(c) Only opto-out connected

(d) Both opto-isolators connected

Figure 4.9: Effects of opto-isolator on (the unused) pin 2

configuration pin 2 has neither of its opto-isolators connected, and thus the active pins, i.e. pin 0 and 1, are the only ones that are connected to opto-isolators.

In the three following configurations, pin 2 is connected to only opto-in (figure 4.9b), only opto-out (figure 4.9c), and finally both opto-out and opto-in (figure 4.9d). From these graphs it is clear that having more opto-isolators on the circuit, even though they are not actively used to transfer data, causes a loss of amplitude on output signals.

### Current vs potential on material

Without opto-isolators involved, the use of the material is mostly concerned with applying voltages to one side, and reading voltages on the other side. Digital equipment typically interpret voltages without drawing much current. The diode on an opto-isolator such as opto-out, however, needs a certain amount of electric power and hence needs to draw more current. Thus, with the opto-isolator scheme without amplification, currents are to a larger degree flowing through the material. We will see in chapter 5 that current can affect the computational properties of the material.

As both input and output pins are connected to an opto-out, and no assumptions should be made to whether a pin is output or input, the resistance between the diode and ground must be kept sufficiently high so that as little voltage as possible is leaked to ground from the material through the input pins, but low enough not to lose voltage on the output diodes. In the former case, input signals

could lose so much voltage that they would not be able to propagate through a high resistance material. Additionally, any pins intended to be high impedance would in fact function more like a drain. In the latter case, however, the loss of voltage on the opto-out diodes could become so large that no wave amplitude would be large enough to be transmitted through it.

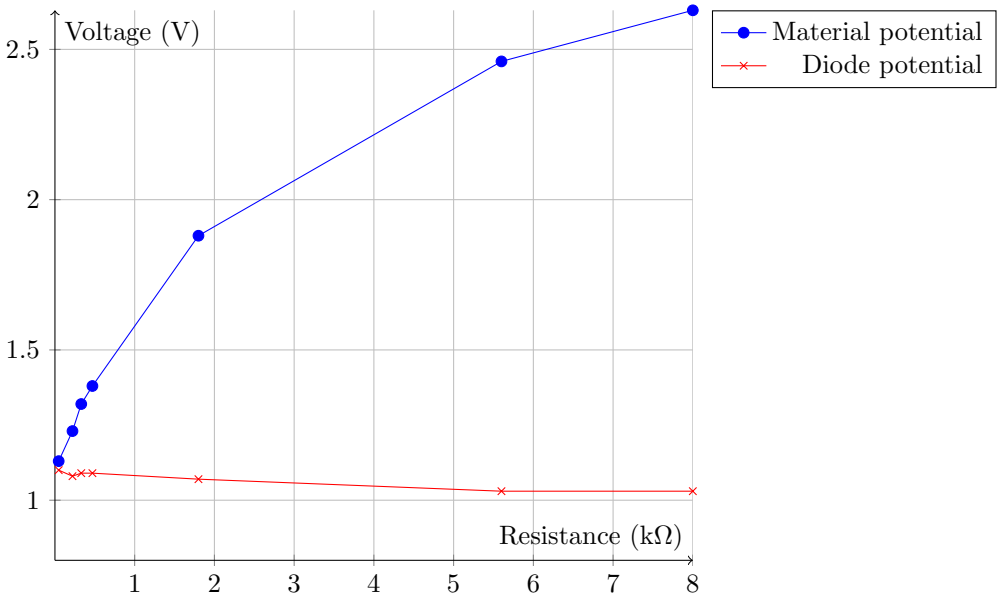



Figure 4.10: opto-out resistance

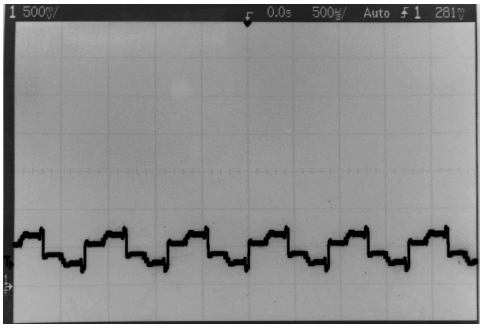
Figure 4.10 shows the relationship between the resistance of said resistor and the two discussed potential differences. The blue plots show the difference between the material pin and ground. The red plots show the difference between the anode and cathode on the opto-out diode. We see here that while increasing the resistance gives smaller decrease in material voltage, it also decreases the voltage on the diode and hence the ability to deliver the signal to the Mecobo-side circuit. Unfortunately, with this amount of pins and resilience to statically assign pins as either input or output, it was not possible to achieve a sufficiently high diode potential so that outputs could be transmitted.

### 4.6.2 Amplifying the output signal

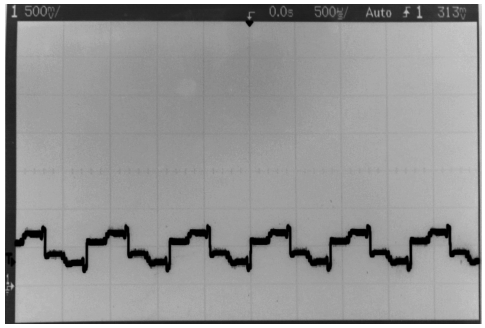
We have established that the amplitude of most waves coming out of the material are too low for the opto-out opto-isolator to transfer back to the Mecobo circuit. This is partly because of the resistance of the material, but mostly because of the loss of voltage described in section 4.6.1. This is the reason that the amplifier component of figure 4.3 (page 47) was included.

Figure 4.11 shows the effect of this amplifier through four different wave mea-

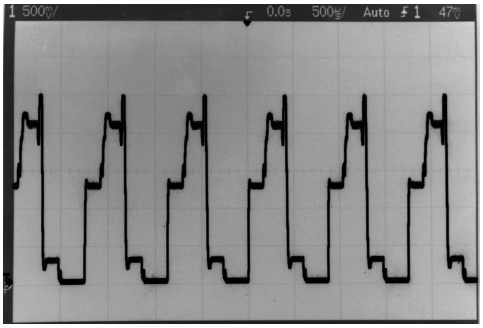
surements. A summary of configurations for the four measurements is shown in figure 4.11e. For reference, a non-amplified signal is included in figure 4.11a. Comparing this signal with that of figure 4.11b, we can see that connecting the amplifier to the circuit has no observable effect on the source signal, at least in this small scale. The differences between the non-amplified signal in figure 4.11a and amplified signal in figure 4.11c shows that the crests are amplified from $\sim 750mV$ up to $\sim 2.5V$, and the troughs from $\sim 200mV$ down to $0V$. Figure 4.11d shows the amplified signal after it has been translated through opto-out. Note that this process inverts the signal. This is the actual input signal into Mecobo and, after being interpreted digitally and inverted, is what is ultimately seen by the GA. Because the amplified amplitude is sufficiently high, this signal can represent the actual material wave.



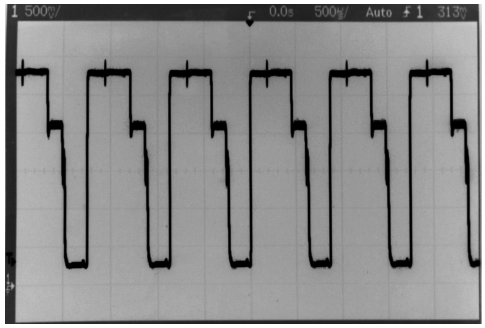
(a) No amplifier connected



(b) Pre-amplification



(c) Post-amplification

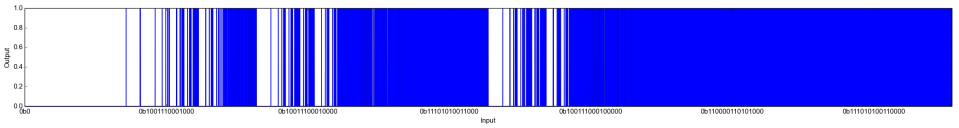


(d) Post-amplification Mecobo-side

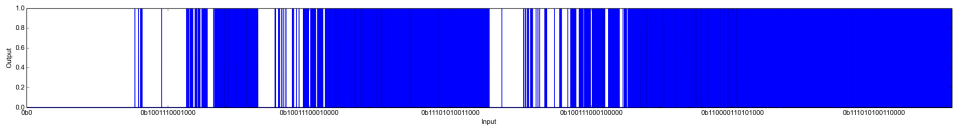
| Figure | Measure point | Amplifier connected |
|---|---|---|
| 4.11a | Material pin | No |
| 4.11b | Material pin (amplifier input) | Yes |
| 4.11c | Amplifier output | Yes |
| 4.11d | opto-out on Mecobo-side circuit | Yes |

(e) Summary of configuration

Figure 4.11: Material-side output amplifier effects




(a) Only output pin amplified




(b) Input pins 0, 7, and 8 also amplified

Figure 4.12: Exhaustive sweeps with amplifiers on input pins

### 4.6.3 Amplifier's effect on computational properties

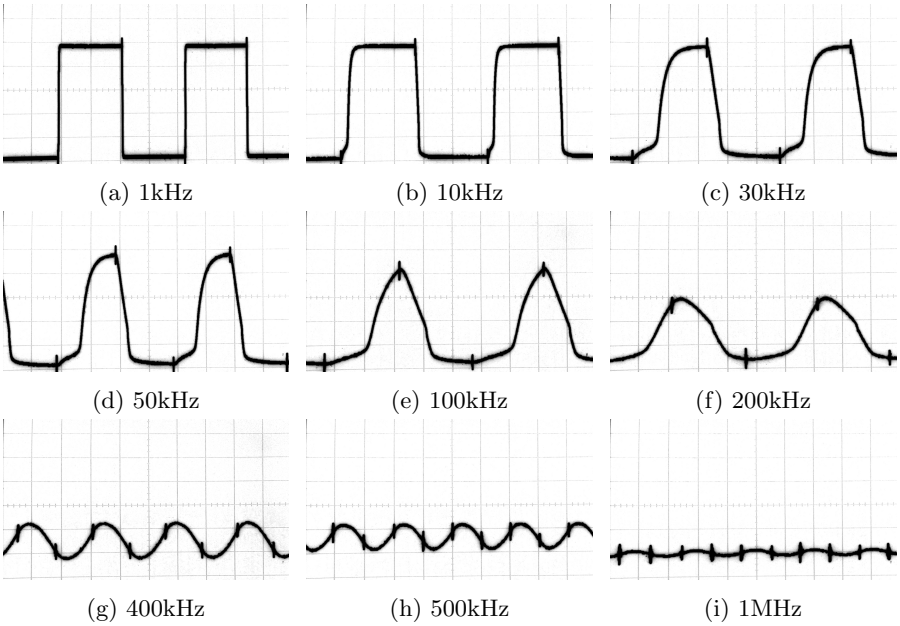
Figure 4.12 shows an example of how input pins with amplifiers connected, though unused, can affect an exhaustive sweep for one output pin. In both sweeps, pin 15 is the (amplified) output pin, and all other pins are used as input. The X axis of the graph represents the input vectors (configuration and input combined), with the far left having all input pins to zero, and the far right all ones. The Y axis represents the output value which is always zero or one. An output value of "1" is shown as a blue vertical line, while the white background means "0". Figure 4.12a shows the negative control with only the output pin amplified, while figure 4.12b shows the same configuration but with pins 0, 7, and 8 amplified as well.

The above results shows that the computational properties have slightly changed upon adding amplifiers. As we have seen in section 4.6, using opto-isolators without amplification resulted in a higher current flow. However, because an ideal operational amplifier has a very high input impedance, we should expect that it now causes only a very small amount of current to flow through the material. Though the computational properties are different with and without amplification, we suspect the effect is smaller when amplification is included.

### 4.6.4 Frequency limitations on material input

The limitations of the switching characteristics of the opto-isolotators restricts the maximum frequency of input waves to and from optowall. Figure 4.13 illustrates this effect with oscilloscope photographs of nine different input frequencies of square waves from Mecobo through opto-in, measured on the corresponding material pin. The square form of the waves where found to deteriorate towards a sinus wave as the frequency increased. Additionally, starting at around 100kHz (figure 4.13e), the amplitude began to decrease until barely present at 1MHz (figure 4.13i).

*Some* loss of amplitude on input pins may be somewhat counteracted by increasing the amplification on the output. However, a similar effect should be expected on the output opto-isolators regardless of amplification, and so an output wave



(a) 1kHz (b) 10kHz (c) 30kHz

(d) 50kHz (e) 100kHz (f) 200kHz

(g) 400kHz (h) 500kHz (i) 1MHz

Figure 4.13: Opto-isolator's effects on square waves of different frequencies

of high frequency would also lose amplitude on the corresponding input[4] pin on Mecobo. If the crests and troughs of this signal becomes below 2V or above 0.8V respectively, the output cannot be observed by Mecobo[7, p. 272]. Additionally, there is no way to counteract the loss of shape on the waveform. In general, as the opto-isolators' ability to transfer information decreases with increased frequency, it is advised to exercise caution regarding the use of high frequencies, especially when these exceed 50kHz.

## 4.7 Determinism of experimental tools

Ultimately, the tools developed in this thesis should be utilized to explore the nature of EiM and the computational material. Initially, there are two ways the tools can be used: Exhaustive sweeps and genetic searches.

### 4.7.1 Exhaustive sweeps

Exhaustive sweeps can map out the computational potential of the material in a crude but relatively deterministic way. An exhaustive sweep probes the material with every possible combination of configuration/input/output sets within a set of restrictions. The restrictions involves the type of signal, the amount of pins, and timing of the probes. We assume, for instance, that there is little point to

[4]From the perspective of Mecobo

running a probe for a lot more or less time than necessary for the output signal to stabilize. Both to keep the time consumption practical, and to be able to present the results in well-known terms, the type of wave is limited to only static voltages and the amount of pins is limited to one output pin, two inputs and the remaining for configuration. The results can then be viewed as basic logic gates such as AND, OR, XOR, etc.

The stability of the output of individual input vectors carries some degree of indeterminism: Some input vectors does not consistently give the same output. However, as one exhaustive sweep consists of $2^{15} \cdot 16$ input vectors, this indeterminism should not be a problem. In general, because of the lack of randomness in the exhaustive sweep, and the relatively stateless property of the carbon nanotubes, exhaustive sweeps are mostly deterministic. However, because of said restrictions, the results are not completely exhaustive and can only give hints of the computational properties. These hints may still be useful, though. Because of their deterministic quality, comparing exhaustive sweeps from different configurations of the experimental platform, e.g. with and without the material interface, the results might reveal interesting properties.

### 4.7.2 Genetic searches

Unlike exhaustive sweeps, it is impractical to attempt to use Genetic Algorithms (GAs) to create a map of the computational potential of a material. However, as GAs are designed to optimize a single well-defined task, they can effectively be utilized to study a few specific functions. For instance can traditional EiM functionalities such as tone discrimination or the traveling salesman problem be evolved both with and without optowall. However, this approach is less deterministic because of the amount of randomness present in the GA.

# Chapter 5

# Experiments

In this chapter we examine the experiments that were performed and their individual results. All the experiments share either one or two common setups, Mecobo with and without the material interface. To avoid having to repeat the description of these for every experiment, they are both documented once in section 5.1. This section also introduces the different material samples used and some common mathematical definitions. The remaining sections of the chapter describes the experiments, where every experiment consists of three sub sections: Goal, setup, and results.

In section 5.2, the exhaustive sweep experiment is the first attempt to show any difference of computational properties, or *functionality*, that the optowall daughterboard causes. When trying to further examine the results of this experiment, stability was found to be a large factor and the next experiment adds stability as a factor in the *stable* exhaustive sweep experiment in section 5.3. Stability was found to be largely improved with the optowall daughterboard, so in section 5.4 the information content of input waves was investigated as a possible cause. Following, in the qualitative and quantitative stability rating experiments, in sections 5.5 and 5.6 respectively, the relationship between gain; stability; and, in the latter, functionality was explored. At the end of the quantitative stability rating experiment, the *nature* of the observed instability was questioned and, in section 5.7, this idea is further explored. Finally, in section 5.8, we present a simple proof-of-concept demonstrating that the presence of evolvability is not removed by the material interface.

## 5.1 Common experimental setups

This section defines the common experimental setups used in the experiments. The individual experiments documented below will refer to these definitions rather than respecify them at every experiment.

### 5.1.1 The control configuration

The control configuration consists of a material slide directly connected to Mecobo. The material slide is connected by 16 wires to a more or less arbitrary set of pins on Mecobo's Header North. The pins are mapped by software so that the material pins from left to right are referred to as pin 0, 1, 2, ..., 15. The FPGA is configured with the "no daughterboard" bitfile, "mecobo_fpga_no_db.bin"[1], and the host server is launched with the `-n` option, which signals that no daughterboards are connected.

The control configuration is the simplest of the experiment setups and involves none of the tools developed in this thesis. It is used mostly as a scientific control to compare with other configurations.

### 5.1.2 The optowall configuration

The optowall configuration consists of Mecobo with the material connected by the material interface, i.e. Mecobo with optowall as daughterboard, and the optoamp extension board connected to optowall.

#### Hardware configuration

The hardware is configured as in figure 4.2 on page 46: The optoamp extension board is connected onto optowall through the "material connectors" and "opto-out GND connector_o2" and o3. All four "opto-out GND connectors" have a jumper connecting their middle pin to AGND. The two of these that connects to optoamp are extended through optoamp and their jumpers are placed onto this extension instead. All four "material connectors", also extended to optoamp, have a jumper vertically on each "in" pin. All four "amp connectors" have vertical jumpers on both in and out for all material pins. These jumper configurations cause the resulting circuit to be as in figure 4.3 on page 47. The large header labeled "HNorth/H-West" on optowall is connected to Mecobo's HeaderNorth. A jumper is placed vertically on DGND and VCC on "HNorth power connector" on optowall so that optowall's Mecobo-side circuit draws power from Mecobo. Not shown in the photograph, on the "power connector" in the lower right corner, a voltage source and ground is connected onto VA+ and AGND respectively. This power supply is separate from Mecobo and its source is 4.0 V.

Unless stated otherwise, the output amplifiers on the optoamp extension board has 0.51V of reference voltage, achieved by having the "RV" resistors $1k\Omega$ and the "RG" resistors $6.8k\Omega$ on the operational amplifiers' voltage dividers. See appendix A for details. In some of the later experiments, this property is recognized as gain.

#### Software configuration

Mecobo's FPGA is loaded with the "mecobo_fpga_optowall.bin" bitfile found under the directory "mecobo_firmware_releases/Mecobo_v3.5_10122012_1" in the `optowall`

[1] This bitfile needed to be recompiled after the 18-09-2014 release, as this release had been incorrectly compiled and had a small bug in the code. The new build and changes in code are available in the `general_fixes` branch of the `NASCENCE/mecobo` repository, see appendix D.

| Identifier | Concentration | Typical resistance |
|---|---|---|
| B08S02 | 1.25% | $50k\Omega$ to $150k\Omega$ |
| B08S04 | 5% | $0.2k\Omega$ to $5k\Omega$ |

Table 5.1: Material properties

branch of the `NASCENCE/mecobo` repository, see appendix D. This bitfile was compiled from the FPGA source code modified in this thesis, and had the `‘define WITH_OPTOWALL` line in both "toplevel.v" and "pincontrol.v". In the former, the `‘define WITH_DB` line was commented/removed.

With this FPGA bitfile, client software can be used as normal. Pin 0 to 15 (inclusive) maps to the material pins. Pin 0 is at the bottom right and pin 15 at the top right of the daughterboard. Note that the pin labels printed on the "material connectors" on the optowall PCB differ from those on the optoamp PCB. In this thesis, the notation on the optoamp PCB has been used consistently. See appendix B for a clarification on labeling and pin mappings. Because the FPGA performs all necessary interfacing to optowall, the server can treat Mecobo as if it has no daughterboards connected. It is thus launched with the `-n` option.

### 5.1.3 Materials

The materials for the experiments are two different samples of the same batch of carbon nanotubes both with a different concentration. The materials where supplied by Durham University at 17th November 2014. The material formulation is described as "single-walled carbon nanotubes mixed with poly(butyl methacrylate) (PBMA) and dissolved in anisole (methoxybenzene)". Their properties are summarized in table 5.1. The identifiers consists of batch number (Bxx) and slide number (Sxx), so both samples are from batch 8.

### 5.1.4 Mathematical definitions

The mean of a list, $x$, of size $n$ is defined as $m(x)$ as in equation 5.1.

$$m(x) = \frac{\sum_{i=0}^{n} x_i}{n} \tag{5.1}$$

## 5.2 Exhaustive sweep experiment

### 5.2.1 Goal

In the exhaustive sweep experiment, we seek to discover simple two-input logic functions in the material. The goal is to establish some crude notion of what the material is able to do in two different configurations: With and without optowall. The results of these experiments may give a clue to whether electrically isolating the material affects the computational properties.

| Probe | Configuration | Gate input | Output |
|---|---|---|---|
| 010011001101010 → 0 | 1001100110101 | 00 | 0 |
| 010011001101011 → 0 | 1001100110101 | 01 | 0 |
| 110011001101010 → 0 | 1001100110101 | 10 | 0 |
| 110011001101011 → 1 | 1001100110101 | 11 | 1 |

Table 5.2: Gate sum extraction with input pins 0 and 14

### 5.2.2 Setup

The setup of this experiment has followed the description of the exhaustive sweep experiment performed by Lykkebø et al. on Mecobo[7]. The experiment began with an exhaustive sweep where all possible 15-bit input vectors where applied onto 15 of the material pins and a response was recorded from the remaining pin. This was done 16 times, each with a different output pin so that all possible pin configurations was used. Each applied input vector and recorded output is from here on referred to as a probe. Every probe consists of a 15 bit "general" input vector and a single bit output. The output is defined as the most common bit in the sampled wave, i.e. "1" or "0". For every probe, the input was applied for a period of 300ms. The output recording began once the input had lasted for 150ms and then lasted for another 149ms.

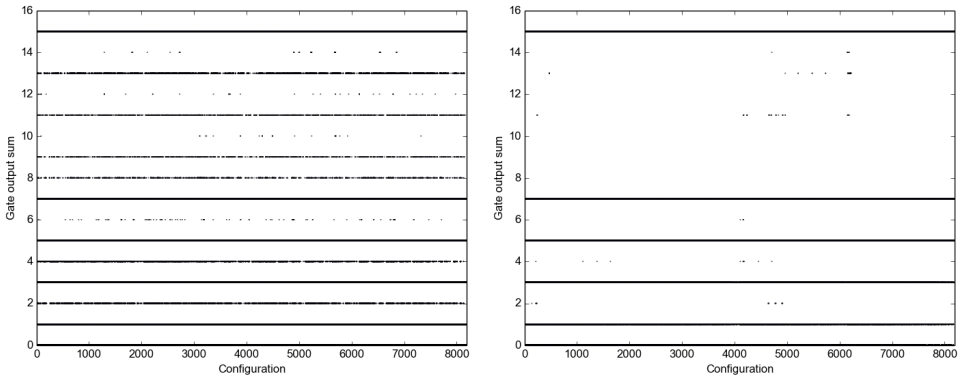
When all probes where exhausted, we used the data to calculate all the possible logic gates that the probes could produce. In this simple experiment, the desired logic gates has one output and two inputs. Three pins was used for this, and the remaining 13 pins was used for configuration. For each set of 15 *general input* pins and 1 output pin from the probes, the logic gates was "extracted" by splitting the general input vectors into two groups, *gate input* and *configuration*. For all the possible variations of gate input pins and configuration pins in all the probes, a tuple of gate input value and output value was stored in a hash map with the corresponding configuration as key. Four such extractions are illustrated in table 5.2. In this table, one configuration value in all its four different states, i.e. with its four different gate input values, have been extracted from four probes. The original probes are shown to the left. Here, we have arbitrarily chosen pins 0 and 14 as input pins, i.e. the leftmost and rightmost bits of the probe's general input vector. These pin numbers are relative to the general input vector of the probe, which consists of any 15 of the 16 available pins from 0 to 15 inclusive. As this process was repeated for all probe data, which consists of all output pin permutations, every possible pin configuration was exhausted.

Every group of four equal configurations represents a logic function. Comparing the gate input with output in table 5.2, we see that it is an AND gate. Every time a logic function is extracted, its *gate output sum* is plotted on a graph. The gate output sum is the decimal representation of the concatenation of all output bits of a logic function. For instance, as illustrated by the output column of table 5.2, the gate output sum of the AND gate is 8, because $1000_2 = 8_{10}$.

In the specific case of time-independent digital logic, this setup should effec-

tively discover all possible logic gates that can be exploited in the material. For each of the material samples, the experiment was performed once with the control configuration (defined in section 5.1.1) and once with the optowall configuration (defined in section 5.1.2).

### 5.2.3 Result



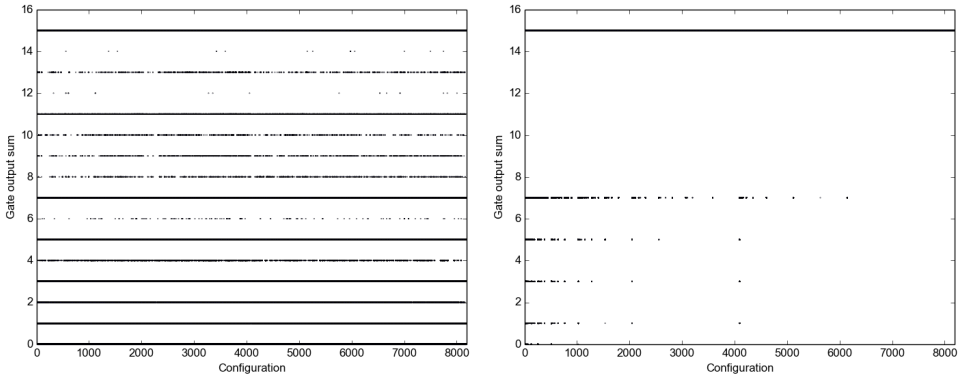

(a) With control configuration

(b) With optowall configuration

Figure 5.1: Gate output sums on material with 1.25% consentration (B08S02)

Figure 5.1 shows the results of the experiment performed on the B08S02 material sample. The X-axis represents configurations and the Y-axis represents the gate sums. Every gate sum is plotted as a single point. As shown in figure 5.1a, all two-input logic gates where abundantly available with the control configuration. With the optowall configuration, however, figure 5.1b shows that the availability of gates is significantly reduced. Some gates, like NOR ($0001_2 = 1_{10}$) and NAND ($0111_2 = 7_{10}$), are still abundantly available. Other gates, like XOR ($0110_2 = 6_{10}$) and OR ($1110_2 = 14_{10}$), are present, but in a lot fewer configurations. Finally, the AND gate ($1000_2 = 8_{10}$) is completely absent.

Somewhat similar results can be observed for the B08S04 material sample, shown in figure 5.2. In the results from the optowall configuration shown in figure 5.2b, however, most of the gate sums are clustered into gate sum 15. In fact, they represent 99.6% of all gate sums. This gate is a TRUE gate, i.e. one that always outputs “1” no matter what input. Because of this, we can derive that most of the probes have observed a “1” as output. Inspecting the configuration vectors of the other gates, most of them contained either one or two “1”s, the rest being “0”s. Because of this, and because the B08S04 material sample has a lower typical resistance, we can conclude that in this particular configuration, the threshold for outputting a “1” is very low. We note for now, that the balance of this threshold effect can likely be regulated by the amount of gain on the optoamp board. We




(a) With control configuration

(b) With optowall configuration

Figure 5.2: Gate output sums on material with 5% consentration (B08S04)

will revisit the effect of gain in the qualitative stability rating experiment of section 5.5.

## 5.3 Stable exhaustive sweep experiment

In an attempt to deepen our understanding of the results of the above experiment, we tried re-create certain gates in the material in order to study their waves at an oscilloscope. This was done by extracting the configuration from some of the more rare gates produced with the optowall configuration, apply this to the material and then send a binary count from 0 to $11_2$ as a wave into the two input pins. In this process, we quickly discovered that we were not able to recreate any of the rare gates. Even when trying one input at a time, as was done in the exhaustive sweep, the gate output sum was often different. As it turned out, stability in the material is a problem in general. This insight lead to the next exhaustive sweep experiment, the *stable* exhaustive sweep.

### 5.3.1 Goal

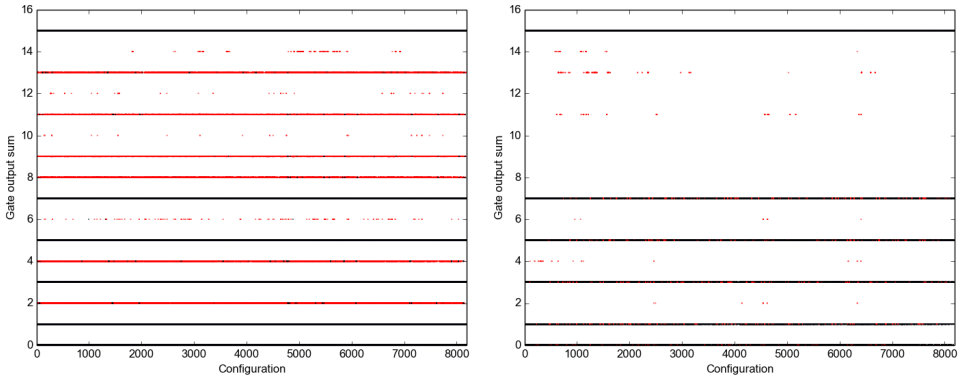
The initial goal of the stable exhaustive sweep experiment was the same as the initial exhaustive sweep experiment (section 5.2), except with more reliable and *stable* logic gates. Additionally, we wanted to compare stability in the different configurations and see whether we could find any patterns in the instability.

### 5.3.2 Setup

The setup was equal to the initial exhaustive sweep experiment, except for the following. Every probe, i.e. appliance of input vector and reading of output, would

be performed three times in a row with a short break in between each. This break was simply the downtime on Mecobo during the delay between commands, where every material pin is put into high impedance state. For every probe, the input is applied for a period of 30ms. The output recording begins once the input has lasted for 5ms and lasts until the input stops. The output samples, now consisting of three separate outputs, would be saved as the arithmetic mean of the outputs where each output is still defined as the majority bit of the recording buffer. This means that stable probes, i.e. probes that exclusively resulted in either ones or zeroes in all three recordings, would have either 1.0 or 0.0 respectively as output. The output of unstable probes would be a floating point number $x$ where $0.0 < x < 1.0$. To calculate the gate output sums, the nearest integer of the output would be used. However, to additionally show other unstable potential gates, a set of unstable gates would be calculated in the same way, except that when an output was unstable, the *inverse* of its nearest integer would be used instead. We name these gates *potential* gates, because they could potentially exist for a brief and probably unpredictable time. They are plotted with red color instead of black.

### 5.3.3 Result




(a) With control configuration

(b) With optowall configuration

Figure 5.3: Stable gate output sums on material with 1.25% consentration (B08S02)

The results for the B08S02 material are shown in figure 5.3. Ignoring the coloring of the graph, it looks a lot like the initial exhaustive sweeps. The colors, however, reveal that the optowall configuration appears much more stable than the control. With the definition of an unstable probe as one whose output is $x$ where $0.0 < x < 1.0$, then the control configuration has 4434 unstable probes while the optowall configuration has 109.

The same was also attempted with the B08S04 material, which is of a higher concentration, and thus a lower typical resistance (see table 5.1). With the optowall configuration, the gate output sum was visually identical as its initial sweep (shown

in figure 5.2b): There were no visible potential gates. In fact, the sweep revealed only a total of 8 unstable probes. In contrast to the stable sweep on B08S02 where seemingly all of the rare gates where potential gates, neither are in B08S04. For the control configuration, the gate output sum plot was very similar as that of B08S02: Except for the simpler gates, most were potential gates and it had a total of 9119 unstable probes. We can thus conclude that the same overall phenomena was present in both materials, though the increase in stability was larger in the B08S04 material. As noted in the results of the first exhaustive sweep experiment, this might also be related to the lower threshold in B08S04.

## 5.4 Comparing information content of material input waves

The stable sweep experiment above suggested that the optowall configuration gives more stable probes than the control configuration. From this, we formed the following hypothesis: *The material input signals contains more noise than the optowall configuration.* In this experiment, we investigated this by comparing the information content of an input wave in the two configurations. This was done by performing a Discrete Fourier Transform (DFT) on both signals.

### 5.4.1 Goal

The goal of this experiment is to obtain proof of the above hypothesis.

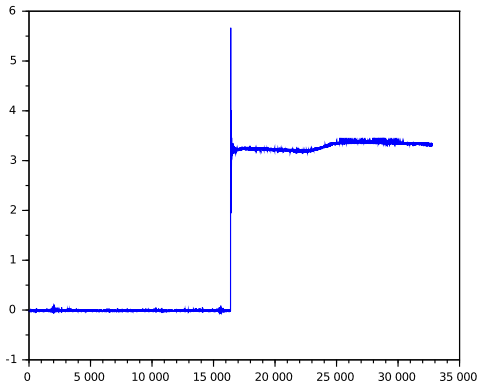
### 5.4.2 Setup

A 1kHz square wave with 50% duty cycle was queued to Mecobo for one of the material pins in the control and optowall configurations. In both configurations, the material was disconnected and the material input pin that delivered the square wave was connected to an Agilent Technologies Logic Analysis System. In the control configuration, said material pin was simply a pin on Mecobo's header north. In the optowall configuration, optowall and optoamp was connected as normal and the material pin was one of optowall's material pins. The high performance logic analyser was used to sample one wavelength of the signal. This sample was then loaded into Scilab[2], where the wave was entered into a fast fourier transform function, `fft`, and then the absolute values of the DFT was plotted to a graph.

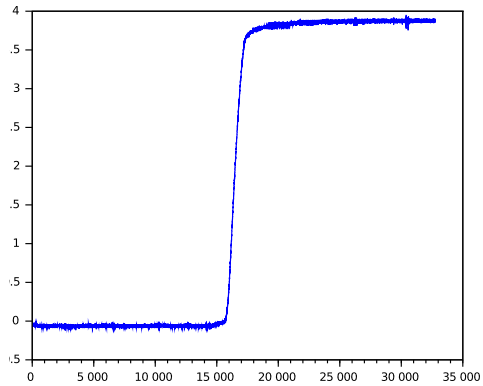
### 5.4.3 Result

The resulting graphs are shown in figure 5.4. For reference, the input waves are shown in figures 5.4a and 5.4b for the control and optowall configurations respectively. A portion of the two DFTs are shown in figures 5.4c and 5.4d. The control configuration in figures 5.4a and 5.4c has a more "correct" square wave. To see

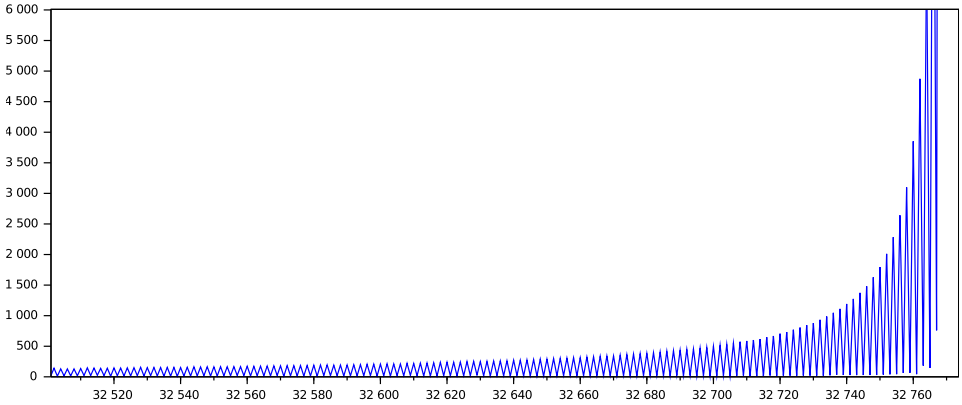
[2]Scilab is a free open-source alternative to MATLAB.



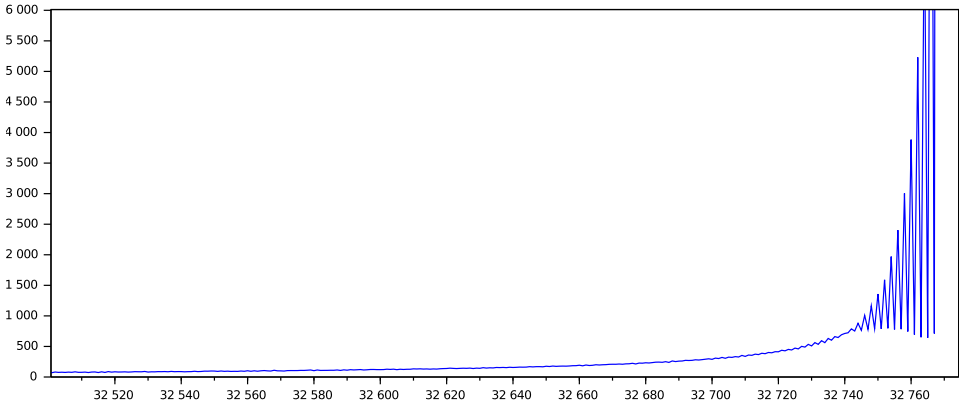

(a) Control configuration




(b) Optowall configuration




(c) Fourier Transform of control configuration




(d) Fourier Transform of optowall configuration

Figure 5.4: Information difference of material input square wave

this we can create a square wave in Scilab by combining a large amount, say $10^6$, of increasing harmonies of sine waves, and plot the DFT of this to a graph in the same way as with the control configuration wave sample. This graph, shown in figure 5.5 is similar to figure 5.4c in that it has a steadily increasing amplitude.

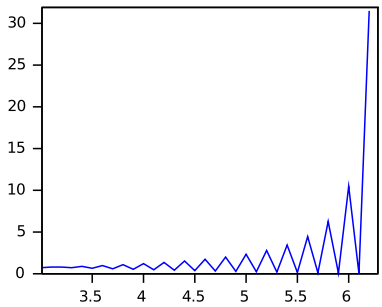



Figure 5.5: DFT of software modulated square wave

Even though the control configuration creates a more correct square wave, it is evident from the DFT plots that the wave of the control configuration contains a significantly larger amount of energy and information than that of the optowall configuration. It is possible that this increased amount of information is a significant source of noise which may explain the material instability in the control configuration. The optowall configuration, producing signals with lower information content, has a significant reduction of instability.

## 5.5 Qualitative stability rating

In the results of the exhaustive sweep experiment of section 5.2, we discovered that an imbalance of threshold effect was present in one of the material samples, and speculated that gain regulation might control this effect. In this experiment, we have studied the effect of gain further.

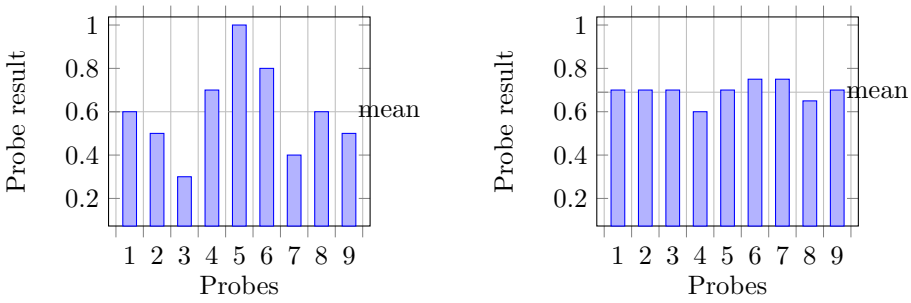
### 5.5.1 Goal

In the Qualitative Stability Rating (QLSR) experiment, we sought to discover a correlation between the amplification gain and stability. We named it *qualitative*, because we performed an in-depth study of one input/output pair.

### 5.5.2 Setup

This experiment was performed on the B08S02 material sample in the optowall configuration. The material was chosen partly because, of the available samples, it had the least extreme values of concentration and typical resistance. Additionally, the stable sweep experiment had revealed it to be more unstable than the others. The input vector $000011111001010_2$ was chosen for pins 1 through 15, while pin 0

was chosen as output pin.[3] This decision was made by looking through the probe logs for the stable sweep experiment and testing some of the probes reported to be unstable: From about 10 arbitrarily chosen unstable probes, the configuration of the one that appeared the least stable was chosen.




(a) Unstable example

(b) Stable example

Figure 5.6: Stablity rating algorithm examples

Each run of the experiment consisted of repeating the chosen probe for a total of $n = 10^4$ times with "blank" probes in between. The blank probes was defined as an input vector of all "0"s on every pin performed for 5ms without recording. The recording buffer for each probe would consist of digital samples performed for a period of about 45ms. This means that the recording buffer from a sine wave would look something like $[0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, \dots]$, having a roughly equal amount of "1"s and "0"s. A constant high, on the other hand, would consist of an array of only "1"s. The result of each probe, $p_i \in P$, was defined as the mean of its recording buffer, e.g. the sine wave would have $p_i \approx 0.5$ and a constant "1" would have $p_i = 1.0$. After all the probes was completed, the mean of all the probes, $m(P)$, was calculated. Figure 5.6a shows an example of nine probes with highly different results. The mean of these example probe results is $m(P) = 0.6$, and many of the results differ largely from the mean. The instability, $I$, is defined as the average distance from $m(P)$, as shown in equation 5.2. For instance, having the recordings averagely close to the mean as in figure 5.6b, indicates stability and $I = 0$ means 100% stability. For clarification, this algorithm is shown in listing 5.1 in Python code, where `recordings` is an array of length $10_4$ containing the floating point results of each probe.

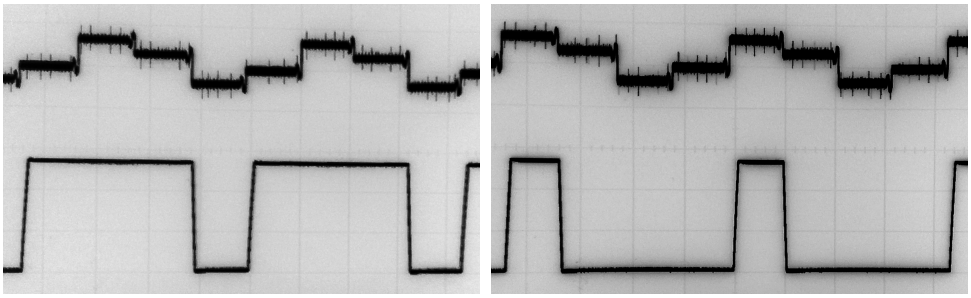
```
mean = sum(recordings) / float(len(recordings))
mean_diffs = map(lambda rec: abs(mean - rec), recordings)
instability = sum(mean_diffs) / float(len(mean_diffs))
```

Listing 5.1: Stability rating algorithm

[3] In the input vector, the rightmost (least significant) bit represents the lowest pin number, in this case pin 1.

$$I = m\left(|m(P) - p_i|\right) \tag{5.2}$$

The experiment was repeated for seven different values of gain on the output pin. The (output) amplifiers of the input pins were not changed. The adjustment of gain was achieved by varying the reference voltage on the amplifier. Figure 5.7 illustrates this effect where the top wave is the amplifier's input wave and the bottom wave is its corresponding output. In figure 5.7a, the reference voltage is lower, which have lowered the threshold for a high voltage output. In other words, gain increases as reference voltage decreases.



(a) 0.36V of reference voltage (b) 0.7V of reference voltage

Figure 5.7: Adjustment of gain

### 5.5.3 Result

The results are shown in figure 5.8, where the instability and mean discussed above are plotted in blue and red color respectively. A pattern that quickly stands out is that the instability appears to be inversely proportional to the mean's distance to 0.0 and 1.0, meaning that a probe is more stable if its recording buffer contains mostly ones or mostly zeroes. This pattern does not, however, tell us much about the material itself but rather about the limitations of digital sampling: When the gain is so low that most samples are zero, or so high that most samples are one, it is likely that an equally noisy wave exists but outside of the thresholds of a digital low or high respectively. The setup of this experiment is thus unable to achieve the goal of finding a pattern between gain and stability.

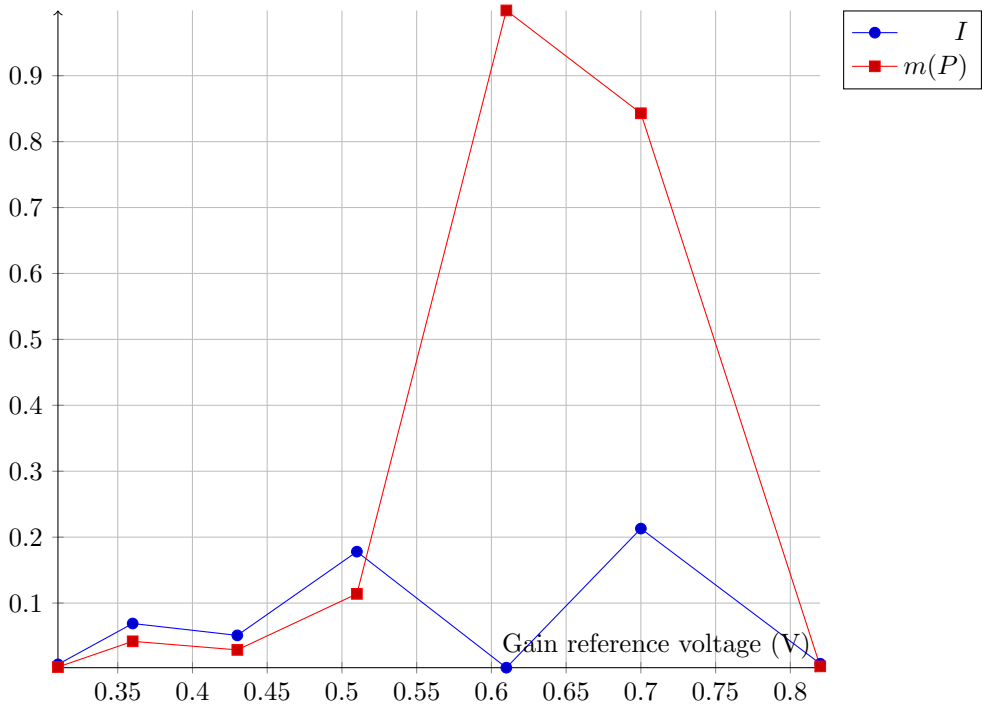



Figure 5.8: Stablity rating

However, there is still some merit to this experiment: In figure 5.8 we can see the mean exponentially rise with the increased gain reference voltage, and then suddenly drop. From this, it is evident that a non-linear function is triggered by only changing gain; every other parameter remained constant.

The high input impedance on the operational amplifier would suggest that very little current should flow through the material. However, when the gain is varied, the input impedance should vary as well. These small changes in impedance should vary the amount of current flowing through. Because we have shown that this effect alone can result in a threshold function in the material, we can conclude that the functionality of the material is indeed sensitive to current.

## 5.6 Quantitative stability rating

### 5.6.1 Goal

In the Quantitative Stability Rating (QNSR) experiment, we initially sought to achieve the same goal as the QLSR experiment, i.e. to determine whether a correlation between gain and stability exists. We call it quantitative because, instead of focusing on a *single* input/output pair, we now determine stability for *every possible* input/output pair. As was shown in the QLSR experiment, the adjustment of

gain moved the parameters of the window for observing the output. Because only one input vector was used, we only managed to show that stability changed because its *observability* changed. In the QNSR experiment, we sought to render this effect statistically insignificant by using every gain setting on all possible input/output pairs, and thus revealing any hidden causes for the change in stability.

The increased scope of this experiment additionally allowed several other variables to be taken into the equation. One of these is functionality, i.e. the computational potential of the material in a given configuration. The first goal is thus to understand how gain affects stability and functionality, and whether a correlation exists between stability and functionality.

Additionally, we hypothesize that the nature of the observed instability is somehow related to the other variables. As a thought experiment, let us consider the material as an RC-network consisting of an even number of input pins and no output pins. When the voltages on input pins are very varied, e.g. half the pins are high and the other half is low, there is a lot going on in the network: Current is flowing through many paths in the network formed by the combinations of high and low voltage pins. However, consider an input vector where all of the pins have equal or very similar voltages. In this case the material is static. There is too little potential difference for any currents to flow. Let us now imagine connecting an output pin into this. With the latter-case input vector, any dynamic functionality observed on the output pin must only be a function of the pin itself. In this experiment, we also wish to see the effect of this factor and its relationship with the other variables. This is made possible by defining and extracting a variable representing the voltage difference of the input vectors of the unstable probes.

### 5.6.2 Setup

The QNSR experiment consists of simply repeating the stable sweep experiment for several different values of gain and then extracting the variables of interest from the output data. The setup for the stable sweep experiment, described in section 5.3.2, was repeated except that gate sums where not plotted. This procedure was repeated for eight different settings of gain. As in the QLSR experiment, gain is defined as the reference voltage in the amplifiers, where lower reference voltages increase the gain. Refer to section 5.5.2 for clarification. Because every output pin was used, each exhaustive sweep had the given gain setting on every output amplifier.

The remaining variables where all extracted from the probe data and are defined in the following paragraphs. Finally these where all plotted to a graph with gain as the X axis, and the extracted variables as Y axis. Some of these extracted variables was scaled using a constant $k$. This was done to be able to plot all variables to the same graph in approximately the same scale. For the variables that are scaled, the *size* of their plot is not important, and we are only interested in the relationship of their trends.

### Stability

Stability, $S$, was defined as the inverse of the total count of unstable probes, $u$, in the given sweep. It is shown in equation 5.3, where $k = 50$. The inverse was chosen for a more intuitive comparison with the other variables: Higher number means more stability. This view works with our non-extreme data. However, as $u$ approaches zero, $S$ approaches infinity. Thus for data sets that have absolute stability, i.e. $u = 0$, the metric fails.

$$S = \frac{1}{u} \cdot k \tag{5.3}$$

### Functionality

To define functionality, the gate sums where computed by the process described in section 5.2.2. The gate sum data was then converted into a list $G$ of size $n = 16$ where every item, $g_i \in G$, contained a count of all instances of the gate sum given by its index $i$. For example, if gate sum 6 (XOR) were available from 10 different configurations, then $g_6 = 10$. Functionality, $F$, was then defined as in equation 5.4: The inverse of the average distance from the mean of $G$. This was scaled by the constant $k = 2 \cdot 10^5$, and $c$ which represents the count of gates that are available in at least one configuration, i.e. $0 \geq c \geq 15$. Another way to view the definition of $F$ is how evenly distributed the gate sums are, where the perfect rating is where all 16 logic gates can be produced by $\frac{1}{16}$ of the total pin configuration and configuration vector combinations. Initially, it was assumed that the functionality rating could be significantly different when including the potential gates, i.e. the red plots in the gate sum graphs of the stable sweep experiment. However, the pattern of the functionality variable did not significantly change when including these. Thus they were excluded, and the gate sum data used was the same as the black plots in the gate sum graphs, i.e. gate sums created from stable probes and the nearest integer of the probe outputs from the unstable probes.

$$F = \frac{1}{\left(\sum_{i=0}^{n} |m(G) - g_i|\right) \div n} \cdot ck \tag{5.4}$$

### Output balance

The additional variable of output balance, $B$, was defined as in equation 5.5, where $n_i$ represents the count of probes that gave $i$ as output. This variable is included to verify that the credibility of the data: If stability would be shown to be directly dependent on this variable's proximity to 1 and 0, then it would be likely that the QNSR and QLSR experiments shared a common flaw, i.e. that the stability improves because the instability of the wave becomes hidden outside the limits of the digital measurement.

$$B = \frac{n_1}{n_1 + n_0} \tag{5.5}$$

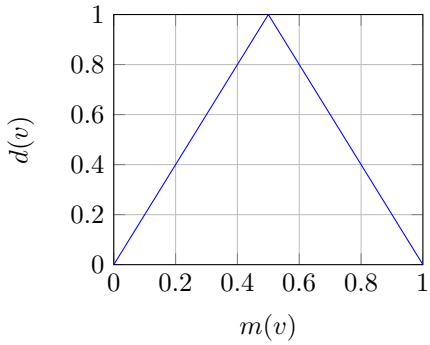



Figure 5.9: Visualization of $d$ function

**Nature of instability**

We define the nature of instability as the voltage difference of the input vectors of unstable probes, or $D$, as shown in equations 5.6. To understand this metric, we first consider the difference of a single input vector, $d(v)$. It was defined as in equation 5.6a, where $v$ is an input vector consisting of 15 binary digits. The $d$ function maps a vector, $v$, into a rating between 0 and 1 representing the total difference of voltage on the vector's input pins. Because every input pin in the vector is either "0" or "1", the largest possible difference of the vector is where exactly half of the pins are "0" and the other half is "1". Inversely, the lowest possible difference is when all the input pins have the same value. It is easiest to visualize the $d$ function with respect to the mean of the vector, $m(v)$. This is shown in figure 5.9. The closer the mean is to 0.5, i.e. the more difference, or variation, exists in the vector, the higher the value of $d$. When there is less difference, i.e. when $m(v)$ is close to either 0 or 1, $d(v)$ is low. In other words, $d$ rewards input vectors that provides large voltage differences in the material, i.e. situations where it is believed that more activity can exist in the material. Inversely, it punishes low differences, i.e. situations where there is less chance for paths of current to emerge.

$$d(v) = 1 - 2 \cdot |m(v) - 0.5| \tag{5.6a}$$

$$D = \frac{\sum_{i=0}^{n} d(v_i)}{n} \tag{5.6b}$$

The purpose of the $D$ variable is to reveal the nature of the observed instability by means of the $d$ function. $D$ was then defined as in equation 5.6b: The mean of the input vector differences, $d(v_i)$, of all $n$ unstable probes $V_u$, where $V_u \subseteq V$ and $V_u = \{v_i \in V : unstable(v_i)\}$.

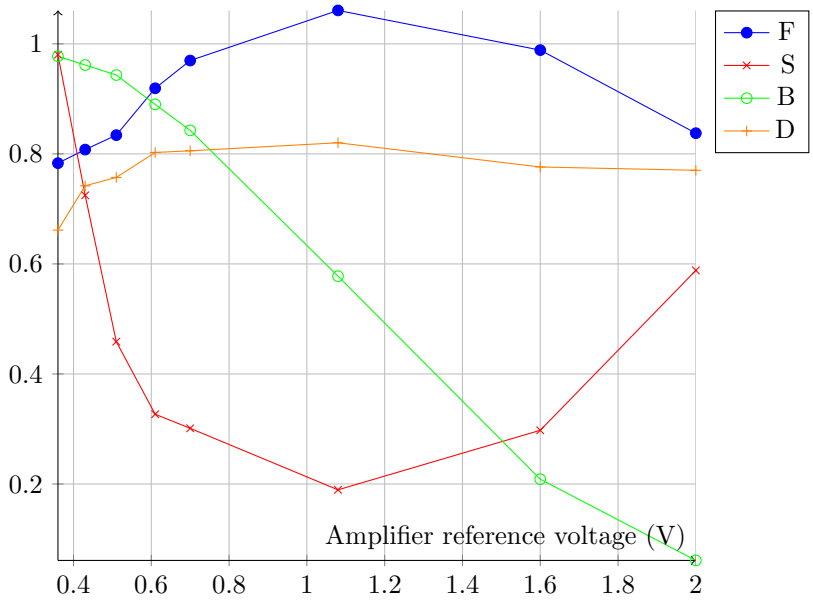



Figure 5.10: Effects of gain

### 5.6.3 Result

The results are shown in figure 5.10. Perhaps the clearest pattern is that **functionality**, $F$ and **stability**, $S$, seems to be inversely proportional. We can view the loss of stability as an increase of chaos. Because the opposite of chaos is order, it might not come as a surprise that functionality, as defined without requirements for stability, would appear to be highest in the chaotic realm. On the other hand, applicable functionality would likely be located at some balance between these, which we will further discuss in section 6.4.

As expected, the **output balance**, $B$, appears to contain *more* "0" outputs when reducing gain, i.e. increasing reference voltage. It appears, however, that $B$ has had a significant effect on the stability: High values of $S$ has, at least to some degree, been caused by the high values of $B$, which implies that the signal was measured as stable only because its instability appeared outside the scope of the digital measurement. This would mean that the actual stability of the signals remained, to some degree, unaffected. From the plot of $B$ it is evident that the X axis in reality represents the movement of the window of observation. There is still merit to adjustment of gain: As long as the signals appears stable within the scope of measurement, they are useful. Also, because gain controls the balance of stability and functionality, we may exploit it to achieve the optimal applicable functionality.

The **nature of the instability**, represented by the input vector difference of the unstable probes, $D$, seems to be very high in most gain settings. This would mean that the instability is mostly caused by high difference input vectors. Also,

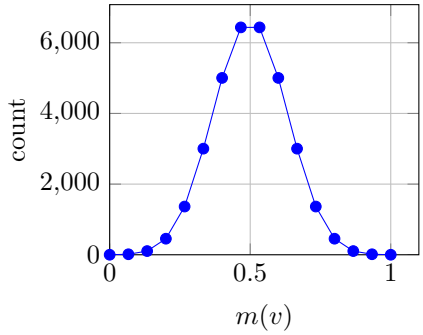



Figure 5.11: Distribution of m(v) for all $2^{15}$ input vectors

it appears to exist a relationship between this variable and stability. If we consider $D$ when stability is low or, equivalently, *in*stability is high, then the instability is averagely caused by high difference input vectors. As the instability decreases, however, the average input vector difference decreases as well. This means that during the process of unstable probes becoming stable, the ones caused by high input difference vectors are the first to become stable. When there are very few unstable probes, they are averagely caused by lower difference input vectors.

It was also hypothesized that rare gates, such as XOR, would be made up of probes with values of $d$ that averagely differed from the more abundant gates such as TRUE. However, no such pattern was found.

## 5.7 Comparison of stability and input vectors

### 5.7.1 Goal

We began studying the nature of unstable probes in the QNSR experiment and, after viewing its result, saw that there existed a relationship between $D$ and stability. In this shorter experiment, we wished to explore this phenomena further, to see if we could gain a clearer and more intuitive understanding of this. To achieve this, we have taken a step back to consider the relationship between the mean of input vectors, $m(v)$, and stability.

### 5.7.2 Setup

As with the QNSR experiment, this experiment was performed by extracting data from stable sweeps performed on several configurations of gain. The X axis was defined as the mean of an input vector, $m(v)$. Recall that an exhaustive sweep uses every possible input vector that can fit into 15 pins. The mean of such an input vector ranges from 0, i.e. the input vector contains only "0"s; to 1, i.e. it contains only "1"s. However, for every value of $m$ where $0 < m(v) < 1$, there are several vectors that maps to the same value. This distribution is shown in figure

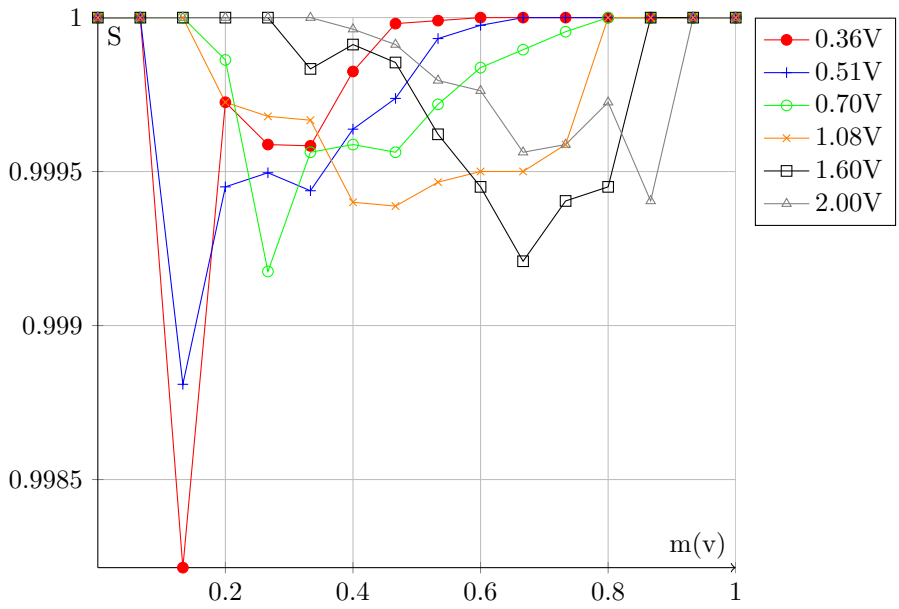



Figure 5.12: Relationship between mean of input vector and stability

5.11, where the Y axis represents the amount of 15 bit vectors mapping to $m(v)$. We view stability, $S$, as a function of $m(v)$: If $V$ is the set of the input vectors of all the probes in an exhaustive sweep, i.e. every possible 15 bit binary vector, then for every value of $x$ in $m(v) = x$, $S(x)$ is defined as the percentage of all the input vectors, $\{v \in V : m(v) = x\}$, that gave an unstable output. For instance, there are 105 different vectors whose mean are all 0.13. If 52 of these gave an unstable output, then a point would be plotted approximately at coordinates $(0.13, 0.5)$, because about 50% are unstable.

### 5.7.3 Result

The results are given in figure 5.12. As was also shown in the QNSR experiment, we can see that stability seems to suffer mostly for medium values of $m(v)$, i.e. high values of $d(v)$. The gain does, however, seem to cause variations of this: With high values of gain, i.e. low reference voltages, the stability decline appears with low values of $d$. This can be seen in the graph as a shift towards the left that is more significant with higher values of gain. This can be explained by the fact that increases in gain causes the threshold for high outputs to be lowered, and thus the instability disappears in the right part, i.e. where $m(v)$ is higher, because the instability in this region has become amplified above the scope of digital measurement. This might also explain the results from the QNSR experiment, which indicated that as stability decreased, more of the instability was caused by high values of $d$, i.e. values of $m(v)$ close to 0.5. It appears that there might exist a

center point where the average instability is located at $m(v) = 0.5$, which appears to approximately be the case with reference voltage 1.08V. This point is also where functionality appeared to be highest in the QNSR experiment.

Another noteworthy aspect about these results is the apparent pattern that is present in every plot. They all seem to start high, then gradually decrease towards a minimum point, $a$, increase to a local maximum, and then decrease to another minimum, $b$, before going up high again. The variation of the pattern seems to occur around three extreme points: The center point, i.e. around 1.08V as mentioned above, the left point, i.e. around 0.36V, and the right point, i.e. around 2.00V.

Around the central extreme point, 1.08V and 1.60V, $b$ is not a minimum, but just has a lower, though positive, derivative. Around the left extreme point, 0.70V to 0.36V, $a$ is a global minimum and $b$ is local. The opposite is seen around the right extreme point, 2.00V, where $a$ is local and $b$ is global minimum.

## 5.8 Genetic search for logic gates

### 5.8.1 Goal

The goal of the GA experiment is to demonstrate that evolvability is present in the optowall configuration. By the use of a GA, we aim to discover a configuration in the material allowing the exploitation of a specific functionality. This approach is the opposite of the exhaustive sweep approach in two major ways. First, instead of seeking an overview of the computational potentials of the material we now wish to narrow the search down to a highly optimized single function. Second, instead of enforcing restrictions to keep the time frame practical, we now try to give the experiment as much freedom as possible. This is possible because of the powerful heuristics of the GA, as discussed in section 2.3. As restrictions are raised, a good GA would most likely discover computational potentials missed by the preceding experiments. However, in this experiment we simply wish to demonstrate the presence of evolvability, so a simpler GA is used.

### 5.8.2 Setup

A genetic search for logic functions on Mecobo has already been successfully attempted by Lykkebø et al.[7]. Because of this, we wished to mimic parts of the setup of their experiment. The similarities includes the genetic representation, fitness function, and the goal of evolving a 2-input XOR logic gate.

The representation was defined as follows. The first two genes assigns input pins and the third assigns output pin. The remaining 13 pins are all assumed to be configuration pins, and the last 13 genes assigns a frequency between 400Hz and 20kHz for every configuration pin.[4] This configuration is visualized in figure 3.12a on page 41. The fitness was measured by applying square waves of the frequencies

[4]The frequency range is smaller than that of the equivalent experiment by Lykkebø et al. because of the frequency limitations on the opto-isolators as discussed in section 4.6.4.

| Input vector | Score weight |
|---|---|
| 00 | 0.3 |
| 01 | 0.5 |
| 10 | 0.5 |
| 11 | 1.15 |

Table 5.3: Score weighting for different input vectors

given by the configuration genes into the configuration pins and then separately applying all four two bit inputs into the input pins while sampling responses on the output pin for 50ms at 100kHz. For all the four outputs, a score between 0 and 1 was awarded according to how close it came to the expected response for its input vector. Additionally, this value was weighted by a constant indicating the difficulty of its corresponding input/output pair, as given in table 5.3. The fitness was defined as the sum of these four scores and thus the perfect fitness would be: $1 \cdot 0.3 + 1 \cdot 0.5 + 1 \cdot 0.5 + 1 \cdot 0.1.15 = 2.45$

Differing from the experiment of Lykkebø et. al. was the usage of a much simpler selection and reproduction scheme: First, a population of $n = 5$ individuals was randomly generated. To create the next generation, the fittest individual was selected and 5 mutated copies of it was inserted into the next generation. Each GA run consisted of a variable amount of generations, and was manually stopped at a more or less arbitrary point.

All GA runs was performed with the material interface, i.e. in the optowall configuration.

### 5.8.3 Result

Figure 5.13 shows the evolution of fitness for an XOR gate for three gain settings. The highest fitness achieved after about 13 thousand generations was $\sim 2.39$ with the least extreme amount of gain (1.60V). The high gain run (1.08V) achieved a fitness of $\sim 1.86$. The lowest gain setting (2V) had the worst performance at fitness 1.70 after about 50 thousand generations. Note that these different runs were performed for a variable amount of generations. The GA used is not ideal, so we did not manage to evolve a perfect gate. As a perfect gate would have a fitness of 2.45, we achieved 97% fitness at the best. This result does, however, successfully demonstrate that evolvability is still present in the material when interfacing it with the optowall daughterboard.

We note for now that the gain setting of 1.60V is probably more ideal than the others for applicable functionality. We will see in section 6.4 that the variance in success for these different gain settings supports a model of how gain should be chosen in general.

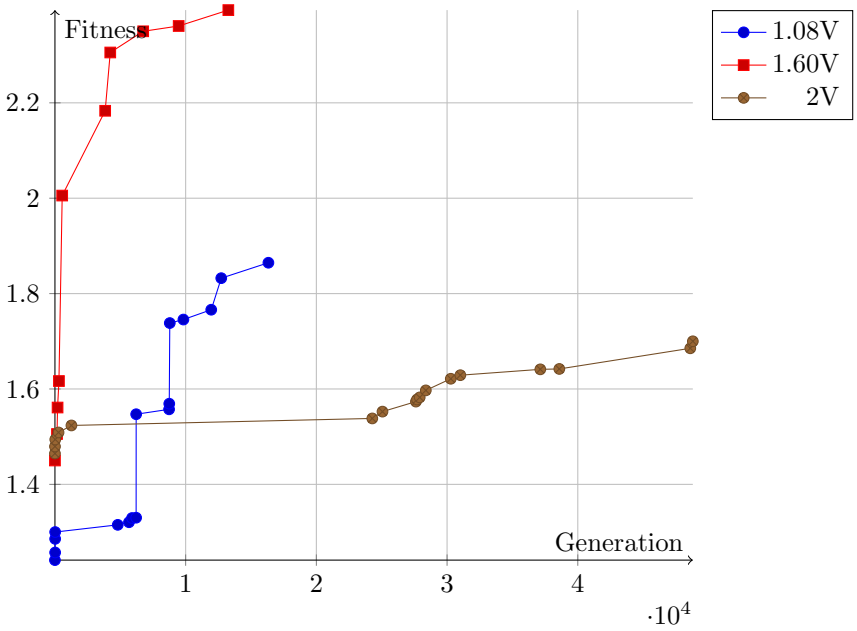



Figure 5.13: Evolution of fitness

# Chapter 6

# Results and Discussion

This chapter summarizes and discusses the results acquired from the experiments in chapter 5. The results can be divided into four major findings: The first, discussed in section 6.1, demonstrates that the method of signal delivery for the material has a significant effect on the computational properties and stability. The second, discussed in section 6.2, shows that non-linear functionality can be triggered in the material by the adjustment of current. The third, discussed in section 6.3, reveals the relationships between functionality and stability, and glimpses the nature of instability itself. Lastly, in section 6.4, we will discuss applicability and how to find the "sweet spot" for the best functionality. We will also, in section 6.5, take the opportunity to make some suggestions of where this research could go in the future.

## 6.1 Method of signal delivery

In previous work there has been variations to how the output signal, i.e. the signal coming out of the computational medium, is driven. In Thompson's evolvable FPGA[4], the output signal was driven by the computational medium itself, i.e. by amplifiers in the FPGA. In Harding and Miller's LCEM[6], in contrast, the output was driven by the input signals, giving the computational medium a seemingly passive role. Similarly, NASCENCE's initial Mecobo experiments[7] may be viewed as having the output driven by input signals. In the initial experiments of this thesis, we have changed the method of signal delivery from being driven by a complex FPGA into being driven by the much simpler opto-isolator circuit. By comparing the former and latter configurations, we have shown the method of signal delivery to have significant effects on the computational properties of the FPMA as a whole. This means that the computational material cannot be viewed as an isolated passive component. The method of signal delivery matters.

In the exhaustive sweep experiments we initially demonstrated that isolating the material from Mecobo greatly increases the stability. We were able to determine this by defining every probe as either stable or unstable and then simply counting

all the unstable probes. We also utilized the concept of gate output sums[7] to plot the existence of all two-input logic gates. By using this technique, we could view the stability of various gates by dividing them into two groups: The normal gates, i.e. the group of gates that could be obtained from the output of stable probes and the most common output of unstable probes; and the unstable potential gates, i.e. the group of gates that were obtained from at least one unstable probe whose output was defined as the least common it observed. By plotting these groups in different colors and comparing with the equivalent plots without this distinction, we could see that many of the configurations previously observed to produce difficult gates, e.g. AND, OR, and XOR, were in fact highly unstable. We had seen that the material interface caused the instability to decline greatly, however it seemed that it had taken functionality with it: The remaining functionality was greatly reduced, and the small amount of remaining configurations that could produce difficult gates where mostly unstable. Despite of this, however, using a GA, we did manage to evolve one of the most difficult gates, XOR, to a fitness of 97%. This demonstrates that the material interface does not remove the presence of evolvability.

By examining the information content in the signals generated from Mecobo and the material interface, we found that the former contained much more information. We note this as a possible reason for why stability appeared lower.

## 6.2 Functionality caused by adjustment of current

In the QLSR experiment, we noticed that while increasing gain for a single input vector, the outputs initially rose as expected, but then suddenly dropped. Because this non-linear behaviour was observed while only changing gain, which slightly alters the flow of current in the material, we were able to conclude that non-linear functionality can be triggered in the material by the adjustment of current alone.

## 6.3 Relationship between stability and functionality

In the later experiments, the main focus was to identify the relationship between various variables in the material, most notably stability and functionality. This was done by making small adjustments to the method of signal delivery, i.e. adjustment of gain in the output amplifiers. This variable would not only move the window of observation of an output signal, but also cause subtle changes to the amount of current flowing through the material for a given input signal. It was quickly found that stability and functionality are inversely proportional. If we were to find the best configuration in which to do stable and interesting computation, we would likely have to find an optimal balance between these. We also found that we were able to use gain to control what we have called “the nature of the instability”.

In the highest values of functionality, the instability seemed to be evenly distributed over many different input vector differences. In figure 5.12 (page 79), the distribution for the highest amount of gain, i.e. reference voltage 0.36V which also

gives the lowest amount of functionality, seems to be clustered towards the left. This means that for the unstable probes, their input vectors had a low difference, or little variation, and most input bits are "0". By decreasing gain into 1.08V of reference voltage, this distribution is much more even and centered around the most varied input vectors, i.e. $m(v) = 0.5$. In other words, when there is more functionality, the instability tends to occur in high difference input vectors. As we believe that high differences on input vectors will cause more activity in the material, it is interesting that functionality seems to be highest when the instability occurs mostly by these input vectors.

## 6.4 Finding the "sweet spot"

With the fact that functionality and stability appears to be inversely proportional, a trade-off is needed to find the optimal applicable functionality, or "sweet spot". In the comparison of stability and input vectors in section 5.7, we identified three extreme points, center, left, and right. The gain setting of 1.08V appeared to have the highest distribution and most centered curve in the $m(v)$ vs $S$ plot in figure 5.12 (page 79). The significance of this center point was also illustrated in the results of the QNSR experiment, presented in figure 5.10 (page 77), where we saw that $max(F)$ and $min(S)$ occurred at approximately this gain setting.

The other two extreme points are caused by relatively extreme values of gain. In the QNSR experiment, they where shown to appear approximately at $min(S)$ and $max(F)$, the opposite of the center point. A possible explanation of this is that we have simply moved the window of observation away from the dynamic and chaotic behaviour of the material. As both of these extreme points show little promise for applicable stable functionality, we propose that there are two "sweet spots" balanced between these three extreme points.

In Cellular Automata, which may be considered a simplified model of a complex system, Chris Langton has suggested that computation, or applicable functionality, must be richest at the "edge of chaos"[24]. The idea here is that a complex system can have two extreme states of highly ordered and highly disordered dynamics, and that useful computation must occur at the transition between these. By the terms of this thesis, we may view the former state as $max(S)$ and $min(F)$ and the latter as $min(S)$ and $max(F)$. Given the correctness of such a view, our balanced points can be considered the edge of chaos and the optimal setting for applicable functionality.

In the GA experiment of section 5.8, we attempted to evolve an XOR gate in different settings of gain. The worst performing GA runs was the gain values of 1.08V and 2.00V, located around the center and right extreme points respectively. The best performance occurred with the gain setting 1.60V, which is somewhat balanced between these two extremes. Looking to the results of the QNSR experiment in figure 5.10, this point has a better balance of functionality and stability. Though the results from these GA runs are not of the highest determinism, as discussed in section 4.7, they seem to support the claim that such "balanced points" has the highest degree of applicable functionality.

## 6.5 Future work

For further exploration of the interface between the material and surrounding electronics, further development of the material interface would be beneficial. Preferably, the adjustment of gain should be possible through software. In the simplest sense, adjustment of gain adjust the measurement characteristics of the interface. As material samples can never be exactly identical, different samples will have different characteristics. In future commercial applications, e.g. when FPMAs are sold as off-the-shelf components, such an automatic control of the material interface might contribute to counteract the effect of these differences. For instance may a lower concentration material sample require more gain than another higher concentration sample of the same material batch. This could be achieved by using separate opto-isolators to adjust the amplifier's reference voltage as illustrated by the simple example circuit in figure 6.1. Here, an opto-isolator is controlled by an analogue pin on Mecobo, assuming such a pin exists. By setting some constant voltage on this pin, the phototransistor on the other side could proportionally open up the flow from VA+, asserting a reference voltage onto the amplifier proportional to the voltage on the analogue pin.

As the experiments in chapter 5 have shown that adjustment of gain can also trigger useful functionality, software adjustment of gain might also be exploited by a GA. This would give the GA even more freedom to discover inherent computational properties, and could have a positive effect on its ability to perform EiM.

A second improvement for the material interface would be to support analogue signals both for material input and output. This would raise the computational paradigm from the digital into the analogue.

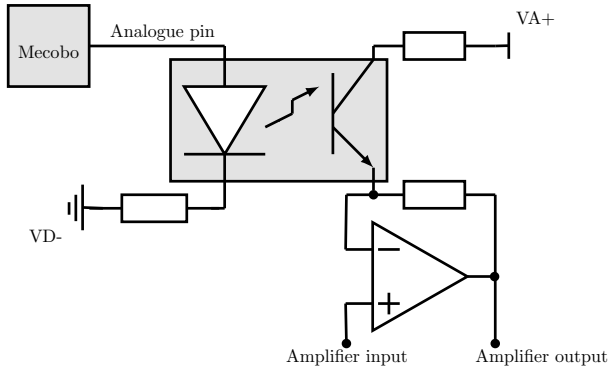



Figure 6.1: Software gain control

# Chapter 7

# Conclusion

In this thesis we initially sat out to explore the implications of electrically separating materials from the complicated circuitry that exploits it. To do this, we developed a material interface that transfers all input and output signals for the material by light instead of electrons on a wire. We quickly discovered that this had significant effects on the computational properties of the material, most notably by stability and functionality. During the design process of the material interface, we realized the necessity of amplifiers in order to truly be able to transfer signals. In the experiment process, we realized that the adjustment of gain on these amplifiers proved a useful method to explore the implications of changing the measurement properties of the material interface. By this we were able to demonstrate the relationship between stability, functionality, and, to some degree, the *nature* of the observed instability.

The material interface drastically increased stability, but it may have come at the cost of functionality. In spite of this, we could still demonstrate the presence of evolvability. We also demonstrated that gain had a significant influence on the search landscape for a GA. Additionally, the difference of evolvability in certain settings of gain can be an indication of favorable dynamic regimes, e.g. edge of chaos, which provides enhanced evolvability and applicable functionality.

As this has been the first effort to explore the effects of electrically isolating the material of EiM, we have limited the task to only use digital signals. In spite of this, we believe our results have revealed interesting fundamental aspects of EiM, and that there is merit to this approach. Our strongest conclusion to the work of this thesis is that the material cannot be viewed as a passive component. The method of signal delivery matters. Because of this, we recommend that future research further explores electrically isolating material interfaces, extended to allow automatic gain control and analog signals.

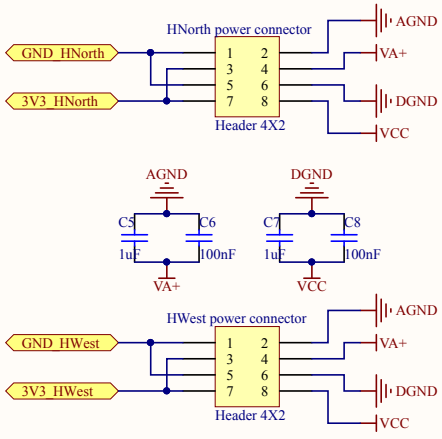
# Appendices

# Appendix A

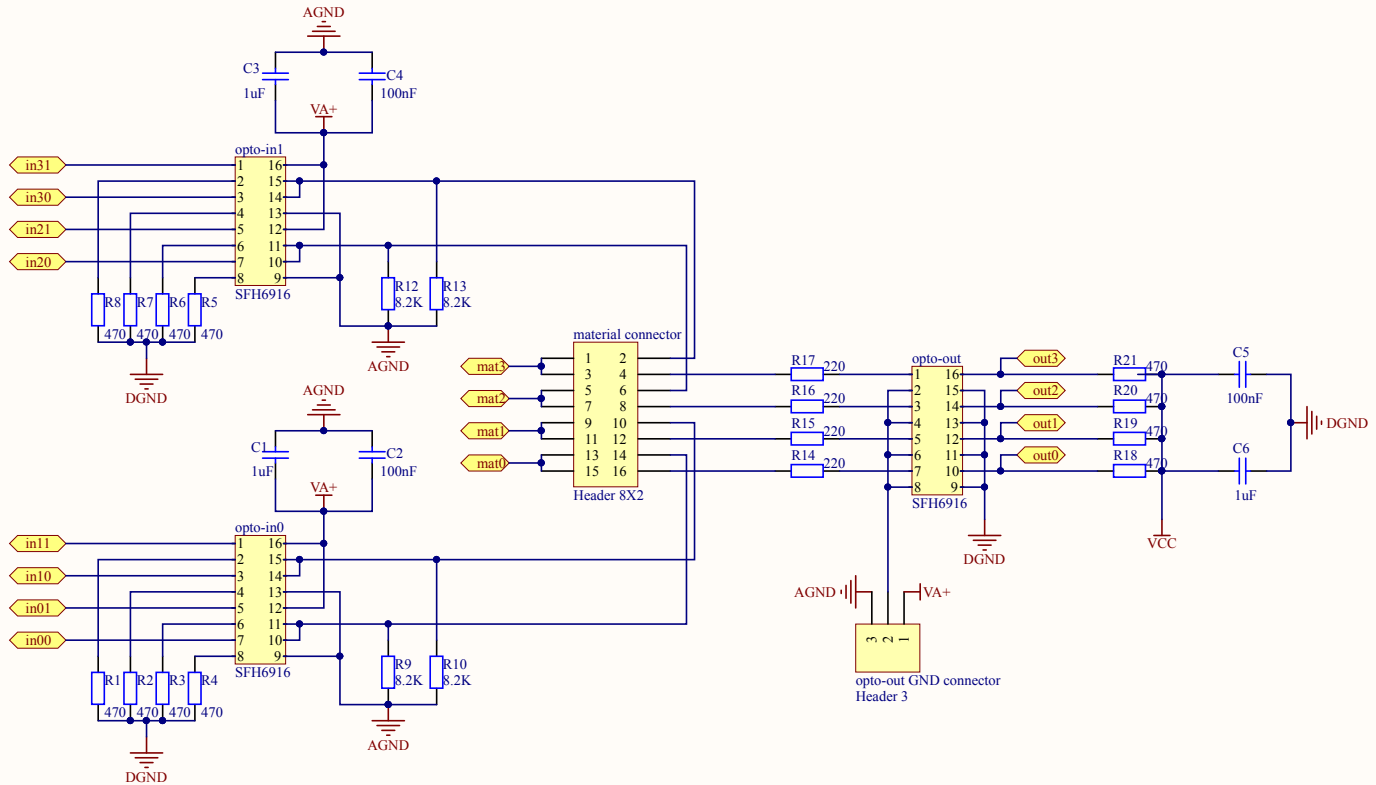
# PCB documents

The following pages contains the schematics, gerber files, and bill of materials for the optowall and optoamp PCBs. The first 6 pages, ending with the bill of materials, are the optowall documents. The remaining are for optoamp. In optowall, the opto-isolators are SFH6916 by Vishay and in optoamp, the operational amplifiers are LM2902M by Texas Instruments.

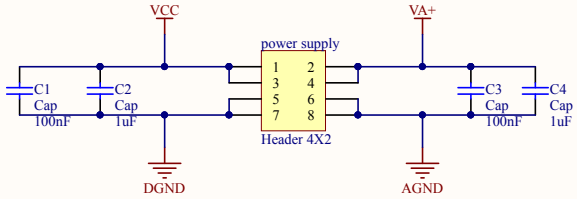
Pin mappings can be deduced from these schematics. However a cleaner overview is included in appendix B.



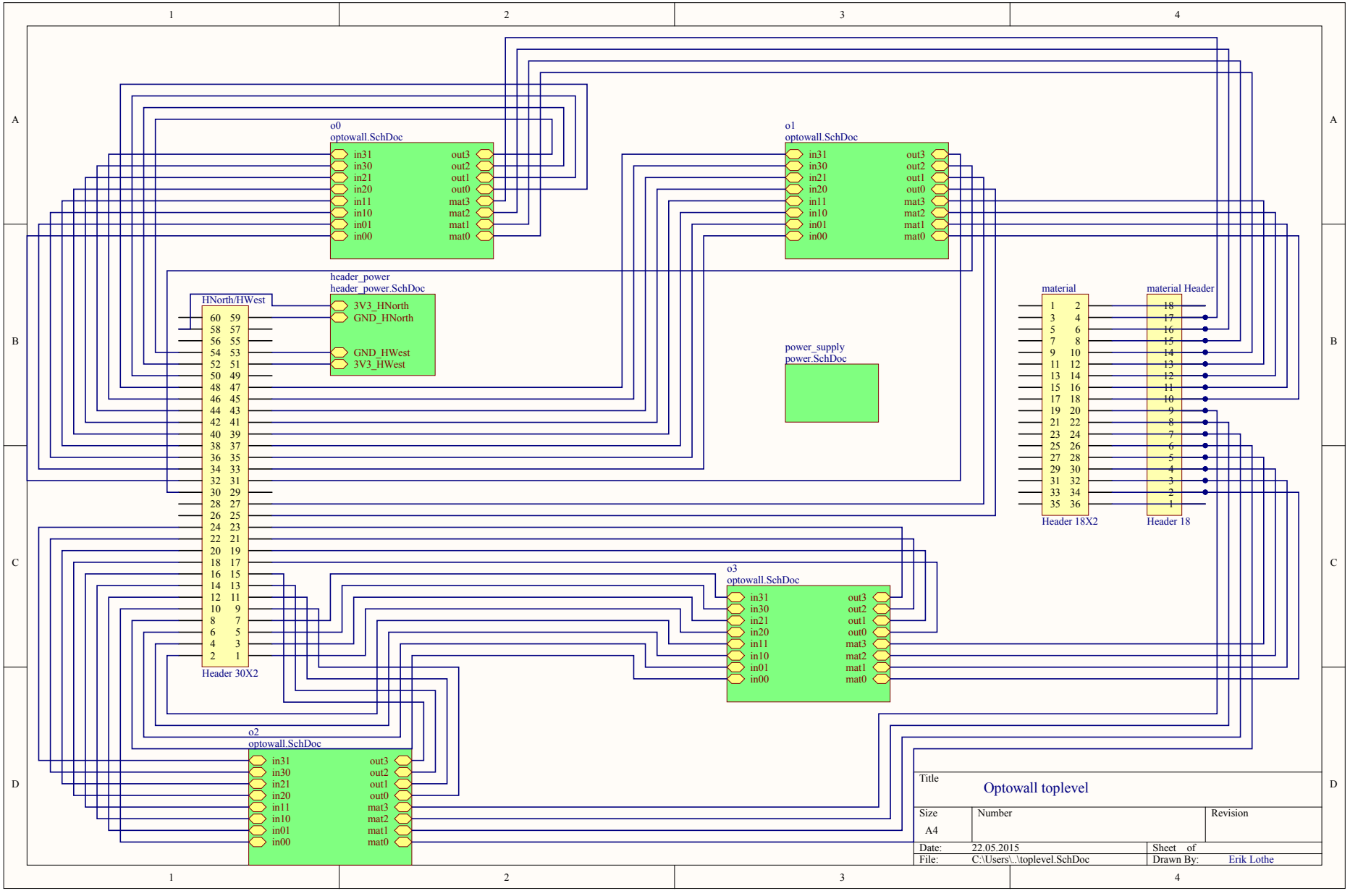

| | | | |
|---|---|---|---|
| Title | Mecobo power supply connector | | |
| Size: A4 | Number: | | Revision: |
| Date: 22.05.2015 | | Sheet of | |
| File: C:\Users\..\header_power.SchDoc | | Drawn By: Erik Lothe | |




| | | |
|---|---|---|
| Title: 4-bit optowall | | |
| Size: A4 | Number: | Revision: |
| Date: 22.05.2015 | | Sheet of |
| File: C:\Users\..\optowall.SchDoc | | Drawn By: Erik Lothe |

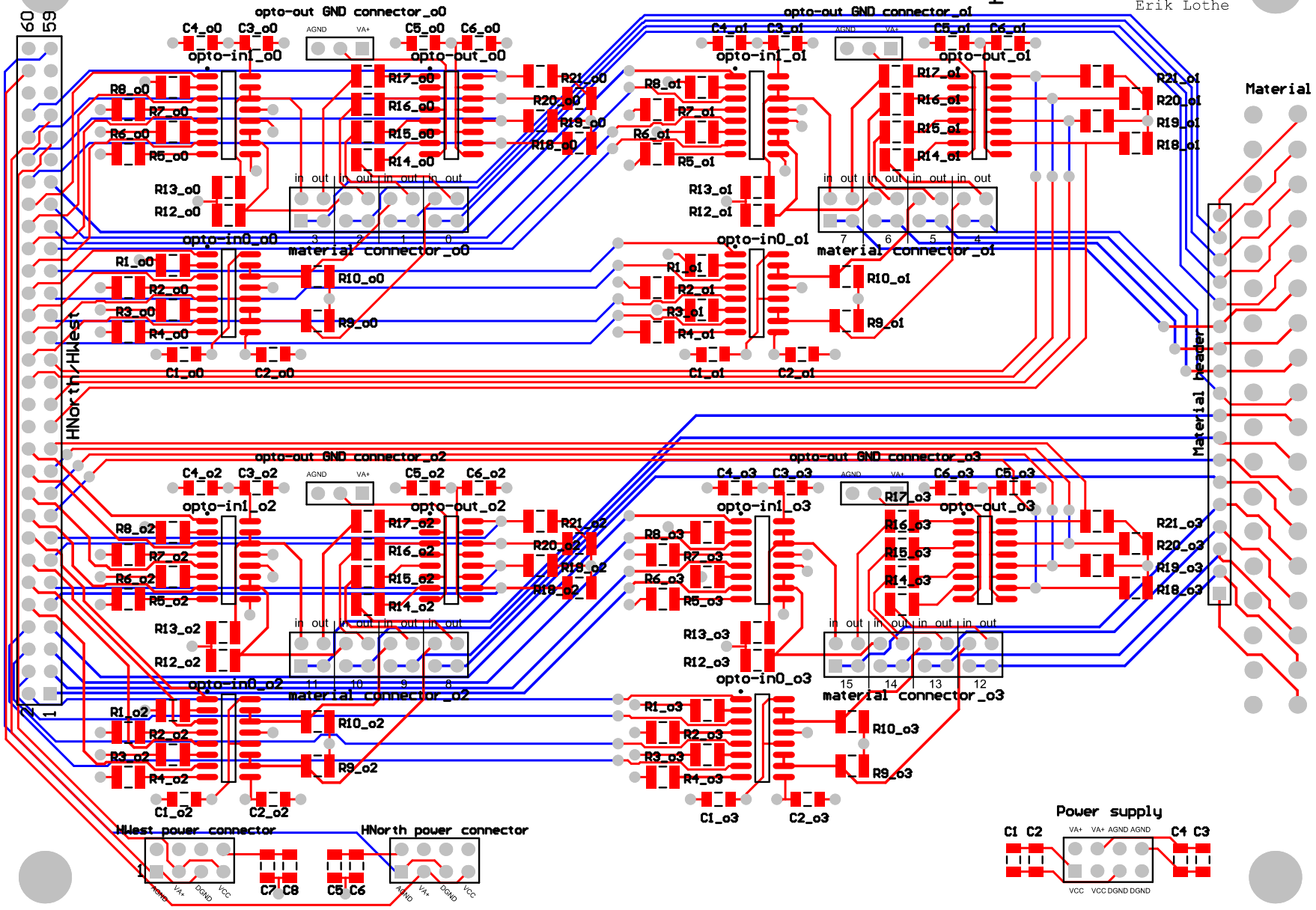



| | | |
|---|---|---|
| Title: Power supply | | |
| Size: A4 | Number: | Revision: |
| Date: 22.05.2015 | | Sheet of |
| File: C:\Users\..\power.SchDoc | | Drawn By: Erik Lothe |

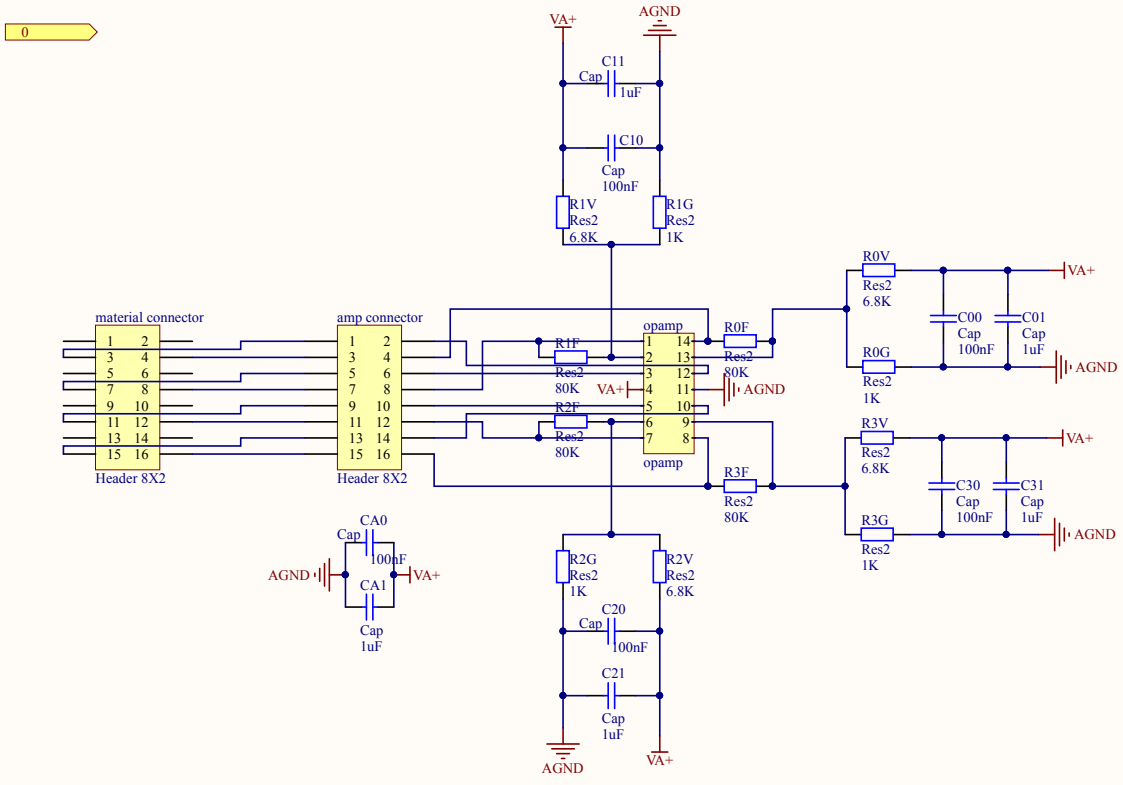

o0
optowall.SchDoc
in31 in30 in21 in20 in11 in10 in01 in00
out3 out2 out1 out0 mat3 mat2 mat1 mat0
o1
optowall.SchDoc
in31 in30 in21 in20 in11 in10 in01 in00
out3 out2 out1 out0 mat3 mat2 mat1 mat0
header_power
header_power.SchDoc
3V3_HNorth
GND_HNorth
GND_HWest
3V3_HWest
HNorth/HWest
60 59
58 57
56 55
54 53
52 51
50 49
48 47
46 45
44 43
42 41
40 39
38 37
36 35
34 33
32 31
30 29
28 27
26 25
24 23
22 21
20 19
18 17
16 15
14 13
12 11
10 9
8 7
6 5
4 3
2 1
Header 30X2
power_supply
power.SchDoc
material
1 2
3 4
5 6
7 8
9 10
11 12
13 14
15 16
17 18
19 20
21 22
23 24
25 26
27 28
29 30
31 32
33 34
35 36
Header 18X2
material Header
18
17
16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
Header 18
o3
optowall.SchDoc
in31 in30 in21 in20 in11 in10 in01 in00
out3 out2 out1 out0 mat3 mat2 mat1 mat0
o2
optowall.SchDoc
in31 in30 in21 in20 in11 in10 in01 in00
out3 out2 out1 out0 mat3 mat2 mat1 mat0
Title
Optowall toplevel
Size
A4
Number
Revision
Date: 22.05.2015
Sheet of
File: C:\Users\..\toplevel.SchDoc
Drawn By: Erik Lothe

# NASCENCE optowall

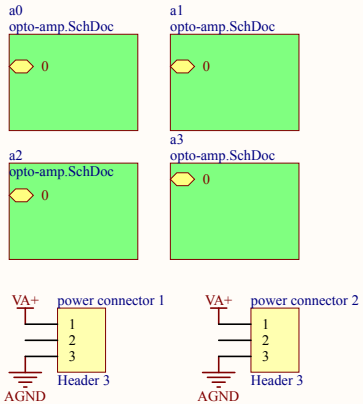
Erik Lothe



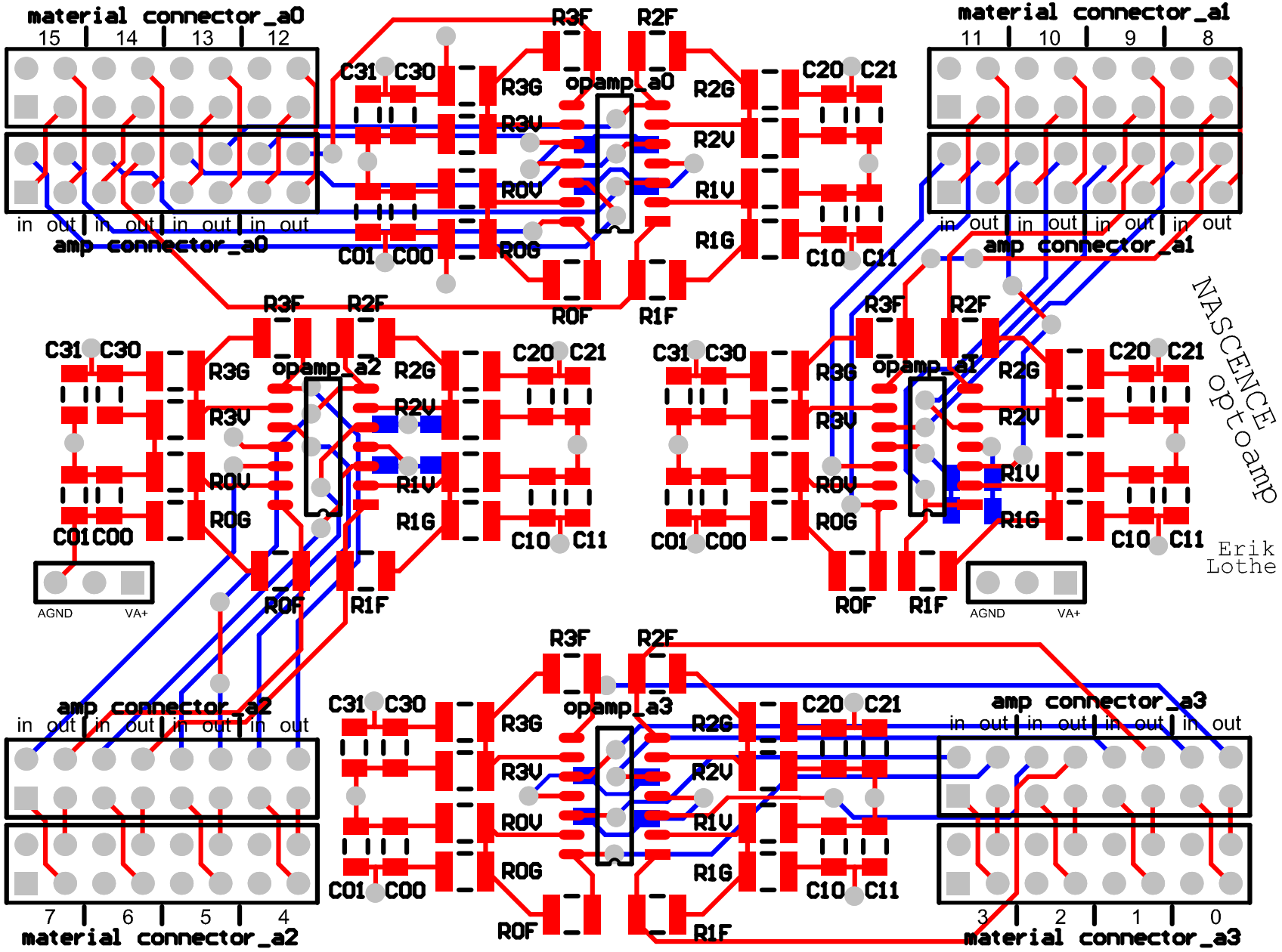
| Comment | Description | Designator | Footprint | LibRef | Quantity |
|---|---|---|---|---|---|
| Cap | Capacitor | C1, C2, C3, C4, C5, C6, C7, C8 | C1206 | Cap | 8 |
| | Capacitor | C1_o0, C1_o1, C1_o2, C1_o3, C2_o0, C2_o1, C2_o2, C2_o3, C3_o0, C3_o1, C3_o2, C3_o3, C4_o0, C4_o1, C4_o2, C4_o3, C5_o0, C5_o1, C5_o2, C5_o3, C6_o0, C6_o1, C6_o2, C6_o3 | C1206 | Cap | 24 |
| Header 4X2 | Header, 4-Pin, Dual row | HNorth power connector, HWest power connector, power supply | HDR2X4 | Header 4X2 | 3 |
| Header 30X2 | Header, 30-Pin, Dual row | HNorth/HWest | HDR2X30 | Header 30X2 | 1 |
| Header 18X2 | Header, 18-Pin, Dual row | material | Material | Header 18X2 | 1 |
| Header 8X2 | Header, 8-Pin, Dual row | material connector_o0, material connector_o1, material connector_o2, material connector_o3 | HDR2X8 | Header 8X2 | 4 |
| Header 18 | Header, 18-Pin | material Header | HDR1X18 | Header 18 | 1 |
| SFH6916 | | opto-in0_o0, opto-in0_o1, opto-in0_o2, opto-in0_o3, opto-in1_o0, opto-in1_o1, opto-in1_o2, opto-in1_o3, opto-out_o0, opto-out_o1, opto-out_o2, opto-out_o3 | SOIC16_M | SFH6916 | 12 |
| Header 3 | Header, 3-Pin | opto-out GND connector_o0, opto-out GND connector_o1, opto-out GND connector_o2, opto-out GND connector_o3 | HDR1X3 | Header 3 | 4 |
| | Resistor | R1_o0, R1_o1, R1_o2, R1 | 14-1210 | Res2 | 80 |




| | | | |
|---|---|---|---|
| Title | 4-bit optoamp | | |
| Size: A4 | Number: | | Revision: |
| Date: 22.05.2015 | | Sheet of | |
| File: C:\Users\..\opto-amp.SchDoc | | Drawn By: Erik Lothe | |




| | | | |
|---|---|---|---|
| Title | optoamp toplevel | | |
| Size A4 | Number | | Revision |
| Date: | 22.05.2015 | Sheet of | |
| File: | C:\Users\..\toplevel.SchDoc | Drawn By: | Erik Lothe |


material connector_a0
15 14 13 12
in out in out in out in out
amp connector_a0
material connector_a1
11 10 9 8
in out in out in out in out
amp connector_a1
material connector_a2
in out in out in out in out
7 6 5 4
material connector_a2
material connector_a3
in out in out in out in out
3 2 1 0
material connector_a3
opamp_a0
opamp_a1
opamp_a2
opamp_a3
R3F R2F R1F R0F
R3G R3V R0V R0G
R2G R2V R1V R1G
C31 C30 C20 C21
C01 C00 C10 C11
AGND VA+
NASCENCE optoamp
Erik Lothe

| Comment | Description | Designator | Footprint | LibRef | Quantity |
|---|---|---|---|---|---|
| Header 8X2 | Header, 8-Pin, Dual row | amp connector_a0, amp connector_a1, amp connector_a2, amp connector_a3, material connector_a0, material connector_a1, material connector_a2, material connector_a3 | HDR2X8 | Header 8X2 | 8 |
| Cap | Capacitor | C00_a0, C00_a1, C00_a2, C00_a3, C01_a0, C01_a1, C01_a2, C01_a3, C10_a0, C10_a1, C10_a2, C10_a3, C11_a0, C11_a1, C11_a2, C11_a3, C20_a0, C20_a1, C20_a2, C20_a3, C21_a0, C21_a1, C21_a2, C21_a3, C30_a0, C30_a1, C30_a2, C30_a3, C31_a0, C31_a1, C31_a2, C31_a3, CA0_a0, CA0_a1, CA0_a2, CA0_a3, CA1_a0, CA1_a1, CA1_a2, CA1_a3 | C1206 | Cap | 40 |
| opamp | | opamp_a0, opamp_a1, opamp_a2, opamp_a3 | SOP14 | opamp | 4 |
| Header 3 | Header, 3-Pin | power connector 1, power connector 2 | HDR1X3 | Header 3 | 2 |
| Res2 | Resistor | R0F_a0, R0F_a1, R0F_a2, | 14-1210 | Res2 | 48 |
| | | | | | |

# Appendix B

# Pin mappings

As the numbering for material pins has admittedly been inconsistent and illogical, pin mappings for the material interface can be somewhat confusing. For convenience and clarification, all pin mapping information is summarized in table B.1 below.

In the table, the **group** number refers to a number $x$ that is printed as o$x$ on optowall (short for opto group $x$) and as a$x$ on optoamp (short for amplifier group $x$). For instance, group 3 is labeled o3 on optowall and a3 on optoamp. The **material pin** columns represents the pin numbers for the pins that connects to the material. These pin numbers have been inconsistently printed on the two PCBs. To clarify, the different pin numbers in a row in the table all refer to the same physical pin. It is only the labeling that differs. In the work of this thesis, including the mappings made in the FPGA source code, the material pin numbers has followed the notation printed on the optoamp PCB. The differing optowall notation is also included in the table. The pin numbers for the **opto-in** and **opto-out** columns all refer to pin numbers on optowall's HNorth/HWest header, shown to the far left in the "Optowall toplevel" schematic in appendix A. These numbers, also printed on the optowall PCB, are consistent with the pin numbers printed on Mecobo's Header North and Header West.

| Group | Material pin | | opto-in | | opto-out |
|---|---|---|---|---|---|
| | optoamp | optowall | in1 | in0 | |
| 0 | 12 | 0 | 34 | 32 | 48 |
| | 13 | 1 | 38 | 36 | 50 |
| | 14 | 2 | 42 | 40 | 52 |
| | 15 | 3 | 46 | 44 | 54 |
| 1 | 8 | 4 | 35 | 33 | 25 |
| | 9 | 5 | 39 | 37 | 27 |
| | 10 | 6 | 43 | 41 | 30 |
| | 11 | 7 | 47 | 45 | 31 |
| 2 | 4 | 8 | 12 | 10 | 9 |
| | 5 | 9 | 16 | 14 | 11 |
| | 6 | 10 | 20 | 18 | 13 |
| | 7 | 11 | 24 | 22 | 15 |
| 3 | 0 | 12 | 6 | 8 | 17 |
| | 1 | 13 | 2 | 4 | 19 |
| | 2 | 14 | 3 | 1 | 21 |
| | 3 | 15 | 7 | 5 | 23 |

Table B.1: Pin mappings

# Appendix C

# Software as experimental tools

This appendix briefly introduces the software developed to aid the experimental process. See appendix D for location of source files.

## C.1 Optoprober

The very first efforts in this thesis consisted of experimenting with various opto-isolators to crudely find out how they worked and how they could be used to isolate a computational material. To do this, an EFM32 Giant Gecko microcontroller on an STK3700 starter kit was used. We developed software for it that can apply square waves and constant voltages to the material through its General Purpose Input/Output (GPIO) pins, both with and without opto-isolators. This software was designed to be quick and easy to use and includes a command line interface accessible through Universal Asynchronous Receiver/Transmitter (UART). The command line supports commands such as performing a single probe with a given value; performing an exhaustive sweep; changing timing of signals; enabling and disabling modes, e.g. debug mode, safety mode, and output inversion mode; etc.

## C.2 Mecoboprober

The mecoboprober software is similar to optoprober, but runs on a PC and interfaces the Mecobo experimental platform through the internet. The structure of mecoboprober is illustrated in figure C.1. At the bottom, `thrift_client.py` is responsible for sending commands to Mecobo through the Remote Procedure Call (RPC) based protocol/language Apache Thrift. A single call to this file will result in commands for one probing, i.e. a binary value to apply to a set of pins in the form of waves or constant voltages, and a set of pins to record an output from. Calls like this are made from the `perform_experiment.py` module. It is

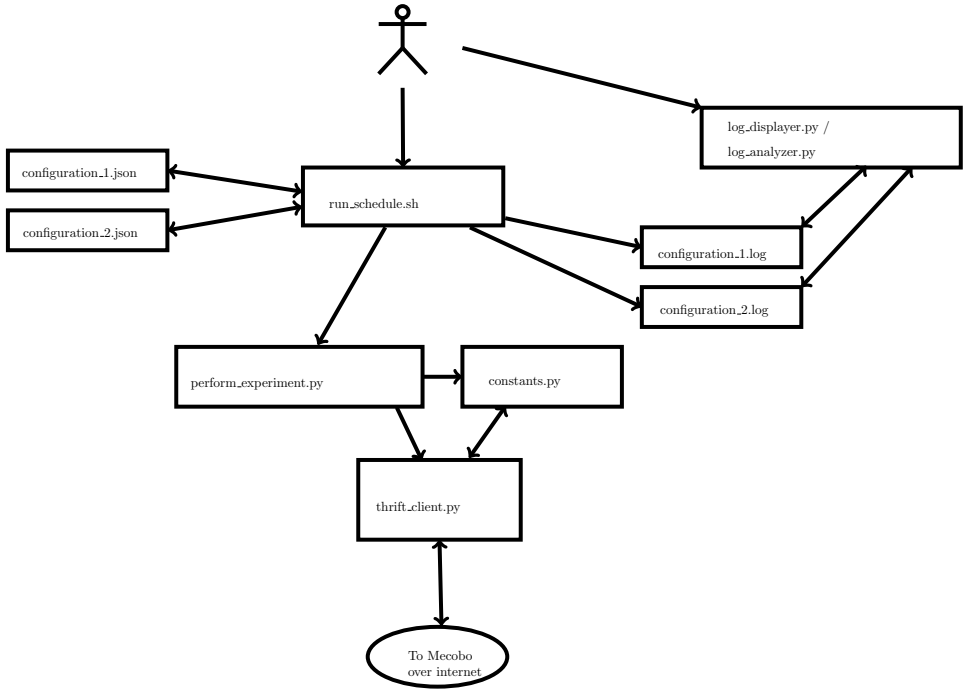



Figure C.1: Software diagram

responsible for a single experiment run, e.g. an exhaustive sweep. An experiment run also has a single set of parameters, e.g the timing of a probe, pin configuration restrictions, and which values to probe. Additionally there are parameters for logging purposes, such as which material is used, name of the FPGA bitfile, whether optowall is used, etc. These are all gathered in the `constants.py` module and can either be loaded statically, or from a configuration file supplied in the command line argument. The `run_schedule.sh` script will detect all configuration files and run each corresponding experiment in turn while redirecting the output into its respective log file. Each log file will include all parameter values and, of course, a list of all input/output pairs it performed.

The user can run a series of experiments by storing a set of configuration files, aided by the `constants.py` module, into a schedule directory and then run the `run_schedule.sh` script. This is useful because a single experiment can last as long as several days. When an experiment is finished, its log file can be opened with the `log_analyzer.py` program. This program allows the user to perform all the tasks on the experiment data that was done in the experiments. Examples of this includes extracting the variables of stability, functionality, input vector differences, output balance, etc. It can also calculate the gate sums and show their scatter as in figure 5.1 on page 65 using the `matplotlib` library. Alternatively, the `log_displayer.py` program can produce a simple X-Y plot as in figure 4.12 on

page 58.

# Appendix D

# Files

This appendix describes the location of the files associated with the work of this thesis. Most of the files are located in a bitbucket repository independent of NASCENCE, the `opto` repository. The rest are modifications to the source code of NASCENCE and is referred to as the `NASCENCE/mecobo` repository.

## D.1 opto repository

The `opto` repository is located at `https://bitbucket.org/89erik/opto/`. It contains all files independent of NASCENCE, such as the developed software, experiment data, and PCB source and output files. The repository contains more information in its `README` file.

## D.2 NASCENCE/mecobo repository

The `NASCENCE/mecobo` repository is located at `https://github.com/89erik/mecobo/`. The changes made for this thesis are located in the `general_fixes` and `optowall` branches. The former contains the re-compilation of the FPGA bitfile for the control configuration (see section 5.1.1), and some minor general fixes to the existing content, i.e. no real new functionality. It has been submitted as a pull request to the equivalent NASCENCE repository. The latter `optowall` branch contains only a single commit: The added FPGA source code and bitfile that integrates the optowall daughterboard with Mecobo. As the changes has diverged significantly with the current HEAD of the NASCENCE repository, it has been submitted as a separate pull request.

# Bibliography


[1] Julian F. Miller and Keith Downing. Evolution in materio: Looking beyond the silicon box. In *Proceedings of NASA/DoD Conference on Evolvable Hardware*, pages 167–176, 2002.

[2] Melanie Mitchell. *Complexity: A Guided Tour*. Oxford University Press, 2011.

[3] Peter Cariani. To evolve an ear: epistemological implications of gordon pask's electrochemical devices. *Systems Research*, (3):19–33, 1993.

[4] Adrian Thompson. An evolved circuit, intrinsic in silicon, entwined with physics, 1996.

[5] Paul Layzell. A new research tool for intrinsic hardware evolution. In *Evolvable Systems: From Biology to Hardware*, volume 1478 of *Lecture Notes in Computer Science*, pages 47–56. Springer Berlin Heidelberg, 1998.

[6] Simon Harding and Julian Francis Miller. Evolution in materio: A tone discriminator in liquid crystal. In *Evolutionary Computation, 2004. CEC2004. Congress on*, volume 2, pages 1800–1807, 2004.

[7] Odd Rune Lykkebø, Simon Harding, Gunnar Tufte, and Julian Miller. Mecobo: A hardware and software platform for in materio evolution. In *Unconventional Computation and Natural Computation - 13th International Conference, UCNC 2014, London, ON, Canada, July 14-18, 2014, Proceedings*, pages 267–279, 2014.

[8] G. E. Moore. Cramming more components onto integrated circuits. *Electronics*, 38(8), 1965.

[9] Marc Duranton, David Black-Schaffer, Koen De Bosschere, and Jonas Maebe. The hipeac vision for advanced computing in horizon 2020, 2013.

[10] David E. Goldberg. *Genetic algorithms in search, optimization, and machine learning*. Addison-Wesley, 1989.

[11] Odd Rune Lykkebø. Design and implementation of a prototype platform for evolution in materio, 2010.

[12] John R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, 1992.

[13] Julian F. Miller, Simon L. Harding, and Gunnar Tufte. Evolution-in-materio: evolving computation in materials. *Evolutionary Intelligence 7*, pages 49–67, 2014.

[14] Alan Turing. On computable numbers, with an application to the entscheidungsproblem. *Proceeding of the London Mathematical Society*, 1936.

[15] M. Morris Mano and Charles R. Kime. *Logic and Computer Design Fundamentals*. Prentice Hall, 2002.

[16] Francis Heylighen. The science of self-organization and adaptivity. In *Knowledge Management, Organizational Intelligence and Learning, and Complexity, in: The Encyclopedia of Life Support Systems, EOLSS*, pages 253–280, 1999.

[17] John H. Holland. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. MIT Press, Cambridge, MA, USA, 1992.

[18] David E. Goldberg. Genetic and evolutionary algorithms come of age. *Commun. ACM*, 37(3):113–119, 1994.

[19] Jason D. Lohn, Gregory S. Hornby, and Derek S. Linden. Human-competitive evolved antennas. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 22:235–247, 2008.

[20] Robert M. May. Simple mathematical models with very complicated dynamics. *Nature*, 261:459–467, 1976.

[21] Stafford Beer. Towards the cybernetic factory. In *Principles of self-organization, Transactions of the Illinois Symposium*, pages 25–89, 1961.

[22] Odd Rune Lykkebø, Stefano Nichele, and Gunnar Tufte. An investigation of square waves for evolution in carbon nanotubes material. In *ECAL 2015, York (in press)*, 2015.

[23] Odd Rune Lykkøbø and Gunnar Tufte. Comparison and evaluation of signal representations for a carbon nanotube computational device. In *2014 IEEE International Conference on Evolvable Systems, ICES 2014, Orlando, FL, USA, December 9-12, 2014*, pages 54–60, 2014.

[24] Chris G. Langton. Computation at the edge of chaos: phase transitions and emergent computation. *Phys. D*, 42(1-3):12–37, 1990.