# COMPUTATION WITH PHOTOCHROMIC MEMORY

Jack Christopher Chaplin, BSc (Hons)

*Thesis submitted to the University of Nottingham*
*for the degree of Doctor of Philosophy*

September 2012

## Abstract

Unconventional computing is an area of research in which novel materials and paradigms are utilised to implement computation and data storage. This includes attempts to embed computation into biological systems, which could allow the observation and modification of living processes. This thesis explores the storage and computational capabilities of a biocompatible light-sensitive (photochromic) molecular switch (NitroBIPS) that has the potential to be embedded into both natural and synthetic biological systems. To achieve this, NitroBIPS was embedded in a (PDMS) polymer matrix and an optomechanical setup was built in order to expose the sample to optical stimulation and record fluorescent emission. NitroBIPS has two stable forms - one fluorescent and one non-fluorescent - and can be switched between the two via illumination with ultraviolet or visible light. By exposing NitroBIPS samples to specific stimulus pulse sequences and recording the intensity of fluorescence emission, data could be stored in registers and logic gates and circuits implemented. In addition, by moving the area of illumination, sub-regions of the sample could be addressed. This enabled parallel registers, Turing machine tapes and elementary cellular automata to be implemented. It has been demonstrated, therefore, that photochromic molecular memory can be used to implement conventional universal computation in an unconventional manner. Furthermore, because registers, Turing machine tapes, logic gates, logic circuits and elementary cellular automata all utilise the same samples and same hardware, it has been shown that photochromic computational devices can be dynamically repurposed. NitroBIPS and related molecules have been shown elsewhere to be capable of modifying many biological processes. This includes inhibiting protein binding, perturbing lipid membranes and binding to DNA in a manner that is dependent on the molecule's form. The implementation of universal computation demonstrated in this thesis could, therefore, be used in combination with these biological manipulations as key components within synthetic biology systems or in order to monitor and control natural biological processes.

**Acknowledgements**

I'd like to thank firstly both my supervisors, Natalio Krasnogor and Noah Russell, for their support, guidance and discussions over the course of my PhD. Though working in an interdisciplinary field with two supervisors from different disciplines was a tricky prospect, it was the belief and encouragement of these two that got me through.

I'd like to thank everyone in the computer science department, the Automated Scheduling, Optimisation and Planning Research Group and the Interdisciplinary Computing and Complex Systems Research Group, with particular thanks to my officemates Lui and German for keeping me company during the long hours at my desk, to the Technical Services Group and particularly Viktor Huddleston for fixing anything and everything that went wrong, and to Deborah Pitchfork, Nick Poxon and Ebru Tasci for their logistical and secretarial support.

I'd like to thank everyone in the Institute of Biophysics, Imaging and Optical Science and the Neurophotonics Lab, with particular thanks to Tim Smith, Kevin Webb and Kelly-Ann Vere for the safety training; basic lessons in chemistry and physics; and simply telling me where things are and what they do. Without the efforts of these three and Noah, I'd have been completely lost in a laboratory. I'd also like to thank those who kept me company in the Biology department, including Alex, Bo, Darren, Katharina, Paul and Solomon.

Outside the university, I'd like to thank my parents for their unconditional love and support, and for fostering my fledging interest in computers and all of science. Thanks to all the teachers and lecturers during my education who encouraged me to pursue my passions. I'd like to thank all my friends, housemates, family, loved ones and especially to my gorgeous Anna Glozier, who all helped me along this journey of discovery (both in science and in self), who were there to celebrate when things went well, and to commiserate when they didn't.

Lastly, I'd like to acknowledge the Engineering and Physical Sciences Research Council and the University of Nottingham's Bridging the Gaps initiative for funding my PhD.

*For everyone.*

# Contents

## List of Figures

## List of Tables

# Nomenclature

**Acronyms and Initialisms**

| | |
|---|---|
| .circ | Logisim Circuit File Format |
| .vi | LabVIEW File Type |
| B | Bleached, a photo-damaged form of NitroBIPS. |
| BNC | Bayonet Neill–Concelman |
| BZ Reaction | Belousov-Zhabotinsky Reaction |
| CA | Cellular Automata / Automaton |
| DAQmx | Data Acquisition Drivers for LabVIEW |
| DIN | Deutsches Institut für Normung (German Institute for Standardization). |
| DNA | Deoxyribonucleic Acid |
| DOM | Document Object Model |
| ECA | Elementary Cellular Automata / Automaton |
| EM | Electromagnetic |
| eNBIPS | Encapsulated NitroBIPS. |
| FRET | Förster Resonance Energy Transfer. Sometimes Fluorescence Resonance Energy Transfer |
| G | LabVIEW Visual Programming Code. |
| I/O | Input / Output |
| I-State | Turing Machine Internal State |
| LabVIEW | Laboratory Virtual Instrumentation Engineering Workbench |
| LED | Light Emitting Diode |
| MC | Merocyanine, the fluorescent form of NitroBIPS. |
| NBIPS | Nitrobenzospiropyran |
| NI | National Instruments. Developers of LabVIEW. |
| NitroBIPS | Nitrobenzospiropyran |
| OD | Optical Density |
| pcFRET | Photochromic Förster Resonance Energy Transfer |
| PDMS | Polydimethylsiloxane |
| SLM | Spatial Light Modulator |
| SP | Spiropyran, the non-fluorescent form of NitroBIPS. |

| | |
|---|---|
| subVI | Sub Virtual Instrument. A LabVIEW subroutine. |
| SUV | Small Unilamellar Vesicle |
| SXGA | Super Extended Graphics Array, 1280×1024 resolution. |
| UV | Ultraviolet |
| UVA | Ultraviolet A, 315-400 nm light. |
| VBT | Value Bound Table |
| VI | Virtual Instrument. A LabVIEW program. |
| VTT | Value Transition Table |
| WXGA | Wide Extended Graphics Array, 1280×800 resolution |
| XML | Extensible Markup Language |

**Symbols**

| | |
|---|---|
| $A$ | Area |
| $B$ | Virtual Boundary |
| $c$ | Speed of light |
| $C$ | Cell array |
| $C$ | Logic circuit |
| $c!$ | Exact gate capacity |
| $C'_n$ | Cell $n$ after the application of F |
| $c+$ | Minimum gate capacity |
| $C_n$ | Cell $n$ |
| $c_x$ | Concentration of $x$. |
| $d_1$ / $d_2$ | Dichroic Mirror 1 / 2 |
| $e$ | Euler's number |
| $erf$ | Error Function |
| $F$ | ECA transition function |
| F | False |
| $g$ | Gate |
| $g*$ | Ordered list of gates |
| $g_{max}$ | ECA generation limit |
| $h$ | Planck’s Constant |
| $i$ | Input node |
| $i*$ | Ordered list of input nodes |
| $id$ | Gate identifier |

| | |
|---|---|
| $I_\lambda$ | Intensity of light of wavelength λ |
| $k_1$ | Decay constant for decolouration |
| $k_2$ | Decay constant for colouration |
| $L$ | Left |
| $l$ | Optical path length |
| $M_{MC*}$ | Measured Fluorescence |
| $M_{offset}$ | Experimental error in Measured Fluorescence |
| $M_{range}$ | Dynamic Range of Fluorescence. |
| $n$ | Index of Refraction |
| $N_A$ | Avogadro's Number |
| $NA$ | Numerical Aperture |
| $N_{ph}$ | Number of photons |
| $O$ | Gate Order |
| $o$ | Output node |
| $o$ | Output pulse |
| $o*$ | Ordered list of output nodes |
| $OD$ | Optical Density |
| $P$ | Power |
| $p*$ | Ordered list of gate pulses |
| $Q$ | Set of Turing machine I-states |
| $R$ | Responsivity |
| $k$ | |
| $\underline{r}$ | Fluorescence Range Bottom Restriction |
| $\overline{r}$ | Fluorescence Range Top Restriction |
| $S$ | Generalised maximum shifts function |
| $\underline{s}$ | Value Lower Bound Skew Factor |
| $\overline{s}$ | Value Upper Bound Skew Factor |
| $t$ | Time |
| $t_{1/2}$ | Half-Life |
| $T$ | Transmittance |
| T | True |
| $V$ | Volume |
| $V_n$ | Register Value $n$ |

| | |
|---|---|
| $\underline{V_n}$ | Register Value Lower Bound |
| $\langle V_n \rangle$ | Register Value Midpoint |
| $\overline{V_n}$ | Register Value Upper Bound |
| $z$ | Number of Standard Deviations |
| Γ | Turing machine alphabet |
| $\delta$ | Turing machine transition function |
| ε | Molar Absorptivity |
| $\theta$ | Angle |
| λ | Wavelength |
| $\sigma$ | Standard Deviation |
| $\Sigma$ | Generalised busy beaver function |
| $\Omega$ | *n*th unique input |
| Ω | Solid Angle |
| Φ | Quantum Efficiency |
| ! | Negation |
| ↑ | Colouration Stimulus |
| ↓ | Decolouration / Excitation Stimulus |
| ¬ | Negation |

# Chapter 1: Introduction

This thesis presents the methods and implementation of computation using memory based on a light sensitive molecular switch with two stable states. By exposing these molecular switches to pulses of light, we can switch the molecule between a fluorescent and non-fluorescent form. Through a combination of light pulses, fluorescence detection, and movable illumination, this thesis shows how to implement registers, Turing machine tapes as parallel registers, logic gates, logic circuits, and elementary cellular automata with these molecular switches.

## 1.1. Background and Motivation

Conventional computing is one of the most important and ubiquitous parts of modern human civilisation, shaping almost all aspects of our lives. Traditionally a computer is an electronic device calculating with binary logic. Unconventional computing challenges this assumption with new paradigms, logics and methods. Since Alan Turing's description of a computational device [Turing (1936)] and the creation of a hierarchy for computational power relative to the Turing machine, we can judge the capabilities of new models of computation and their physical realisation relative to one another.

Key to conventional computing is the logic gate. Boolean functions can be implemented with two-value electrical switching circuits [Shannon (1938)], allowing for the processing of algorithms and mathematics. Modern logic gates are typically constructed from silicon transistors, with miniaturisation being the key challenge to improving execution speeds. Billions of dollars are spent on chip fabrication

techniques, resolving feature sizes as low as 14nm or smaller [Singh (2011)]. A logic circuit is a set of inputs, logic gates, and outputs which implement a more complex logical function than a single logic gate. This allows for a wide variety of combinatorial logic expressions to be implemented with common components.

The use of unconventional materials to implement logic gates and computing in general is an example of the field of unconventional computing. Unconventional computing challenges the norm on several points. Conventional computers are traditionally monolithic black boxes, sealed from their environment, under desks. They require constant sources of power, either from mains or battery. They can be fragile to temperature and mechanical damage when improperly protected. They would be expensive to produce, design and build though economies of scale offset this. Though transistor fabrication technology has kept up with Moore's Law [Moore (1965)], the need to shrink the feature size to maintain progress is rapidly becoming untenable [Zhirnov et al. (2003)] [Keyes (2001)].

Unconventional computing is an extremely diverse field, with research from quantum mechanics [DiVencenzo (1995)] to slime molds [Adamatzky (2007)]. It is not bound to a particular logical paradigm nor to the material with which it is implemented. It may be more or less powerful than traditional computers [Potgieter (2006)], it may be functional in more extreme situations or repair itself [Ramachandran et al. (2012)], it may have dramatically reduced levels of power consumption [Toffoli (1980)] [Vitányi (2005)], it may be as small as a single molecule [Conrad (1985)] [de Silva and McClenaghan (2004)] or it may self assemble from simpler components [Li et al. (2009)], and it may be embedded into materials in our environment [Greenfield (2006)].

Unconventional implementations of logic gates exist as diverse as carbon nanotubes [Bachtold et al. (2001)], quantum dots [Loss and DiVincenzo (1998)], rotaxane molecular architectures [Collier et al. (1999)], prokaryotic transcriptional networks [Silva-Rocha and de Lorenzo (2008)], smart polymers [Pasparakis et al. (2009)], bacteriorhodopsin films [Rao et al. (1996)] and DNA [Elbaz et al. (2010)].

A common issue in unconventional approaches is the implementation of 'wires'; the communication between gates which simulate wires in electronic circuits. Within biology, signals are passed in two ways; as the diffusion of chemical

signals, and as electrical action potentials in neurons. Molecular scale electronics aims to produce electrical components on the molecular scale [Aviram and Ratner (1974)] [Reed et al. (1997)] [Flood et al. (2004)] and supramolecular scale [Schenning et al. (2004)], possibly offering the fast communication of signals. The vast array of chemical reactions can be leveraged as communication methods, where signals are chemicals that react with components, and outputs are more chemical signals or other stimuli. [Balzani et al. (2003)] [Prasanna de Silva and Uchiyama (2007)].

One of the fields of unconventional computing attracting attention is natural computing. Natural computing takes inspiration from the biological world, and applies it to computational fields. It can apply to algorithm design (such as evolutionary algorithms or neural nets), or to the use of natural techniques or materials to build computing devices. The interest in the natural world for inspiration for computational advances is (pardon the pun) a natural one. Biology performs some remarkably complex computation, from the simplest bacteria to the most complex device known to humanity; our brains.

In his landmark paper [Adleman (1994)], Adleman showed how mixing specific DNA sequences in an otherwise unordered and unstructured solution could solve a seven-node Hamiltonian path problem by exploiting the massive parallelisation offered by the large number of interacting DNA strands. Research into leveraging biological computational components such as DNA, proteins, bacteria, neurons and other cells holds promise for several reasons. Firstly, the pedigree of biological components is apparent; they are already complex computational devices in their own right, with regulatory networks and internal structure to maintain homeostasis [Levine and Davidson (2005)]. Secondly, they exist in nature in aggregate, capable of self assembly into colonies or organisms more complex than our most advanced fabrication methods are capable of [Li et al. (2009)], and suited to the communication and sharing of information. Thirdly, the blueprint to many of these components is now modifiable; advances in DNA sequencing and sequence synthesis allow us to create custom DNA strands and transfect these into cells to alter their behaviour [Felgner et al. (1987)] [Menuel et al. (2008)].

The field of synthetic biology seeks to produce custom behaviour in life forms. There are two approaches; bottom-up and top-down. Top-down takes existing life and modifies it to produce new behaviour, but this can cause problems when the existing complicated regulatory networks interfere with the new components in unexpected ways [Silva-Rocha and de Lorenzo (2008)]. Bottom-up starts from simpler chemical elements and seeks to produce a minimal protocell which can then be extended with defined modules.

One approach to bottom-up synthetic biology is liposome logic [Smaldon et al. (2010)]. Liposome logic is the containment of minimally complex biological regulatory networks in liposomic vesicles, designed to compartmentalise functionality and prevent unintended interactions. Communication between components is via molecular diffusion through pores in the vesicles. The creation of reusable common modular components such as logic gates and counters helps control complexity inflation.

Synthetic biology systems can benefit from external control, allowing the user or environment to interact with internal processes. Molecular switches can be used to govern some biological processes [Hille (2001)] [Hammes (2002)] and alter their state and associated properties in response to stimulus. For example, photochromic molecules are molecular switches for which the stimulus is light. Spiropyrans are a class of photochromic molecule that can be used to alter biological processes when bound to other molecules, including protein binding [Sakata et al. (2005)] and membrane perturbation [Ohya et al. (1998)]. As the control of proteins and membranes are crucial to liposome logic, and as optical stimulus is a non-invasive method of interaction with a system, this thesis aims to study and characterise a spiropyran with regards to its own capability as a computing substrate.

## 1.2. Research Goals

The research goals considered in this thesis are the following:

- Investigate the properties and capabilities of a biocompatible molecular switch which could be used to implement a register.
- Implement registers using these molecules, and implement fundamental conventional logic elements as an extension to these registers.

- Extend the fundamental conventional logic elements to implement a universal computing paradigm.

### 1.3. Publications

During the project, a peer-reviewed journal paper, a peer-reviewed extended abstract, two peer-reviewed conference publications in proceedings (one abstract, one long-form), two workshop presentations and three posters were produced. Two additional journal publications are being prepared. Their details are found below:

Peer Reviewed Journal Paper

- J.C. Chaplin, N.A. Russell, and N. Krasnogor. Implementing conventional logic unconventionally: Photochromic molecular populations as registers and logic gates. *Biosystems*, 109:35–51, 2012.

Journal Papers/Letters in Preparation

- J.C. Chaplin, N. Krasnogor and N.A. Russell. Photochromic Molecular Switch Implementations of Turing Machines and Elementary Cellular Automata. *PlosONE*. (Paper)
- J.C. Chaplin, N.A. Russell and N. Krasnogor. Title TBC. *Nature*. (Letter)

Peer Reviewed Extended Abstract

- J.C. Chaplin, N. Krasnogor, and N.A. Russell. The effect of encapsulation on molecular computing efficiency. In *Functional Optical Imaging (FOI),* 2011.

Conference Publications

- J.C. Chaplin, N.A. Russell, and N. Krasnogor. Implementing Conventional Logic Unconventionally: Photochromic Molecules as Registers and Logic Gates. *Proceedings of the First COBRA Workshop on Biological and Chemical Information Technologies*, 2011.
- J.C. Chaplin, N. Krasnogor and N. Russell. Towards a Physical Implementation of P Systems: Photo-switching Molecules as Logic Gates and Registers. *Proceedings of the Third International Workshop on Physics and Computation.* 2010.

Workshop Presentations

- J.C. Chaplin, N.A. Russell, and N. Krasnogor. Computing with Photochromic Molecules. *First COBRA Workshop on Biological and Chemical Information Technologies*, 2011.
- J.C. Chaplin, N. Krasnogor and N. Russell. Towards a Physical Implementation of P Systems: Photo-switching Molecules as Logic Gates and Registers. *The Third International Workshop on Physics and Computation.* 2010.

Posters

- J.C. Chaplin, N. Krasnogor, and N.A. Russell. The effect of encapsulation on molecular computing efficiency. *2011 International Conference on Functional Optical Imaging*, China.
- J.C. Chaplin, N. Krasnogor and N. Russell. Towards a Physical Implementation of P Systems. *2010 Cross Disciplinary Research Showcase,* Nottingham.
- J. Chaplin, N. Krasnogor, N. Russell. Unconventional Computing: Implementing the Unusual. *Bridging the Gaps Interdisciplinary Research Showcase 2009*, Nottingham.

## 1.4. Structure of the Thesis

The *Background Material* chapter presents the context within which the rest of this thesis is considered. It looks at possible methods by which our research goals can be fulfilled, and settles on a likely candidate biocompatible molecule; NitroBIPS, a photochromic spiropyran. *Chapter 3: Computational Model and Programmability* explains the model that underlies our use of photochromic molecules as memory and registers, and also explains how to program the system. *Chapter 4: Methods* details the methods necessary to encapsulate NitroBIPS samples and enact the switching behaviour which forms the framework from which subsequent work draws. *Chapter 5: Photochromic Molecular Registers* goes into more detail on the effect of light pulses on NitroBIPS and details the implementation of data storage in eNBIPS samples as registers, the core component from which subsequent chapters expand. *Chapter 6: Steps Toward Photochromic Molecular Turing Machines* details the implementation of a Turing machine tape using parallel eNBIPS registers. *Chapter 7: Photochromic Molecular Logic Gates* details the expansion of registers to implement logic gates, and details how multiple logic gates

can be executed to form logic circuits. *Chapter 8: Photochromic Molecular Elementary Cellular Automata* details how elementary cellular automata can be implemented with a combination of registers and logic circuits. *Chapter 9: Discussion and Conclusions* concludes the thesis with a summary of research and presents a discussion on possible directions for future research and applications. Figure 1.1 shows the chapters of this thesis and how they relate to one another.

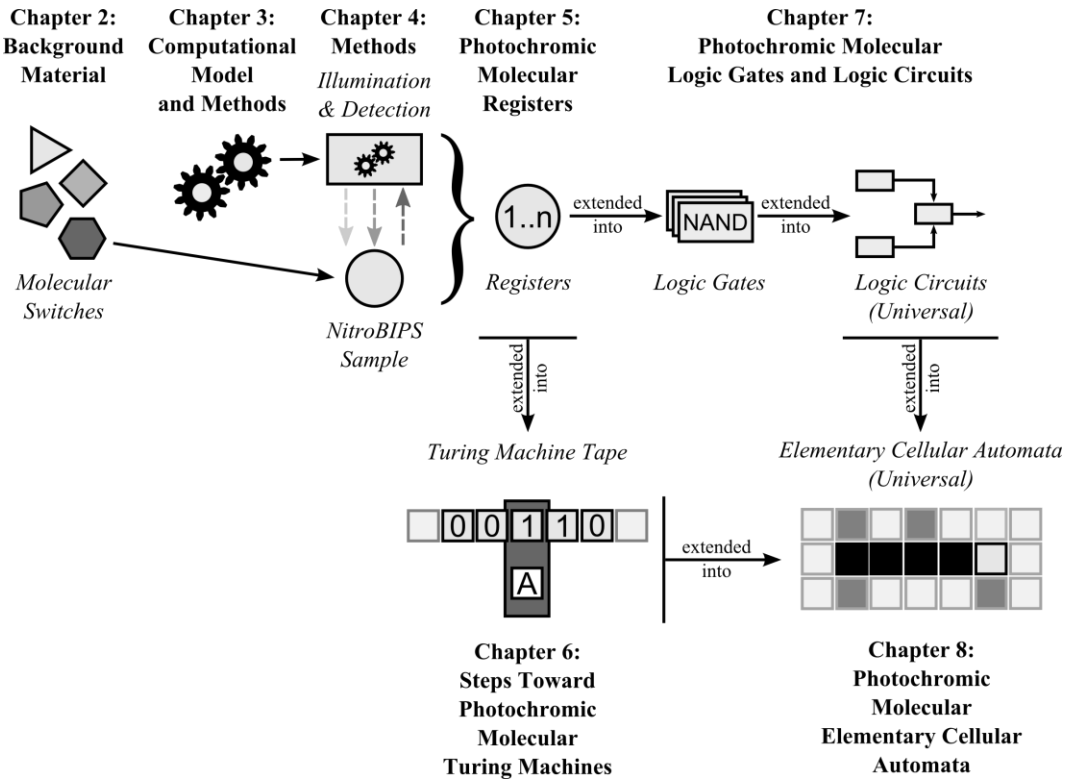



*Figure 1.1: A pictorial representation of the structure of this thesis.*

# Chapter 2: Background Material

## 2.1. Conventional Computing

The idea of using tools to help the process of calculation has existed since man first counted on his fingers. The development of the abacus and a succession of mechanical devices for counting and calculating led to the work of Charles Babbage on the Analytical Machine [Bromley (1982)]. Though never built, the design for this mechanical computer shared many of the features enshrined by John von Neumann in his model of digital computer architecture [von Neumann (1945)][1] [Null and Lobur (2010)], and the more general idea of a stored-program architecture. The stored-program architecture places processing instructions and the working memory within the same storage area, allowing the execution of self-modifying programs.

Attempts to formalise the fledging notion of 'computation' lead to three models of computation; Church's λ-calculus [Church (1932)] [Cardone and Hindley (2009)], Post's machine [Post (1936)] and Turing's a-machine, [Turing (1936)] [Boolos et al. (2002)] now commonly referred to as a Turing machine. λ-calculus takes a mathematical form, similar to a minimal programming language, consisting solely of variables, abstraction symbols and parenthesis to disambiguate statements. Post's machine and Turing machines take a very similar approach, despite being formulated independently. Both represent extremely simple models of mechanical

[1] This incomplete report was distributed and incorrectly treated as published by many when it first appeared in 1945, resulting in some controversy over the attribution of the stored program concept to Von Neumann exclusively.

computation, though Post's representation is more restrictive and is now often considered as a sub-class of Turing machine.

The Turing machine originated as a *Gedankenexperiment*, and was Turing's idea for a hypothetical minimal mechanical computing device that simulated the way human clerks operated in an office during Turing's lifetime. It is visualised as a mechanical device with four primary components, as depicted in *Figure 2.1*;

- A limitless bi-directional one-dimensional *tape*, which is divided into discrete *cells* into which a single symbol from a defined alphabet is stored (typically *0* or *1*). The alphabet contains a default 'blank' symbol which tape cells begin storing. The tape is arbitrarily extendable in either direction such that the Turing machine never runs out of storage, but still only uses a finite amount of space. The input to the system is the pattern of symbols on the tape at the start of execution.
- A movable *head*, which moves relative to the tape, scanning a single cell at a time. The head can move, read the symbol in the current cell, and write new symbols to cells.
- An *internal-state register* which stores the '*I-state*' of the machine. There is a defined start I-state in which the machine begins execution, and a defined halt I-state, which terminates execution. Turing made a strong distinction between the system's '*state of mind'* and its '*state of progress*' through the execution. Hence, the term *internal-state* or *I-state* is used to disambiguate from the machine's state of progress.
- A *transition function*, which is a set of instructions that governs the operation of the machine. Each step, the I-state and the symbol written to the current cell are read, and cross referenced to give three actions, executed in order;
    i. Erase the old symbol and write a new symbol to the cell (this may be the same symbol that was just erased).
    ii. Move the head one cell left or right.
    iii. Update the internal-state register to a new I-state.

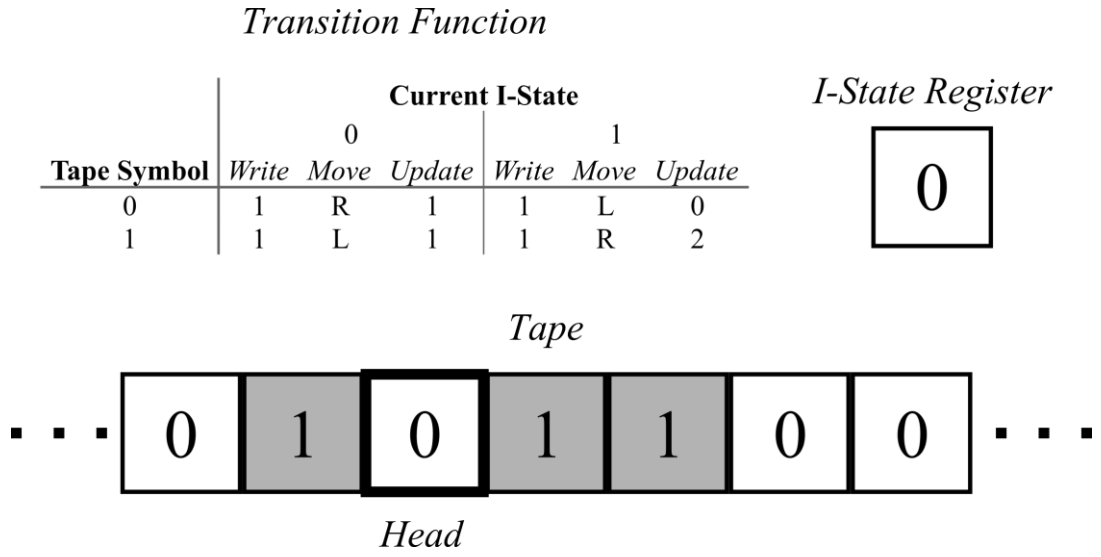



*Figure 2.1: The four components of a classic Turing machine.*

Like λ-calculus and Post's machine, the Turing machine was designed to explore the limitations of what is possible for computers to compute; the definition being in a paper discussing the *Entscheidungsproblem* or '*Decision Problem*' [Turing (1936)]. The Decision Problem asked if it was possible to develop a general algorithm that would take a logical statement, and answer if that statement was provable. Turing reduced this to the halting problem, to which Turing subsequently proved was uncomputable. In doing so, Turing also defined the universal Turing machine, which was the first formal description of a stored-program computer. A universal Turing machine is a single machine which can simulate any other, by taking a description of the simulated Turing machine as the starting state of the tape. This places both the 'program' (the transition table of the simulated machine) and the working memory on the same storage medium, and allows a single machine to process any algorithm and any input into that algorithm.

The two formalisms lend their names to the Church-Turing thesis [Kleene (1967)] [Church et al. (2006)], which states that any computable problem (or algorithm) is computable on a Turing machine. As an algorithm is any problem which can be divided into a finite number of discrete steps, this includes all known mathematical and logical functions. It implies that any computer which can execute algorithms cannot solve problems a Turing machine cannot solve. The Church-Turing thesis has, however, never been formally proven. [Buss et al. (2001)]

The Church-Turing thesis gives us a measure of computational power relative to the Turing machine:

i. *Sub-Turing*: The computing paradigm cannot execute all algorithms a Turing machine can.
ii. *Turing Complete*: The paradigm can compute every Turing-computable algorithm.
iii. *Turing Equivalent*: The paradigm can compute every Turing-computable algorithm, and is proven to be no more powerful.
iv. *Super-Turing:* Sometimes called a hypercomputer, the paradigm can compute algorithms a Turing machine cannot.

Independently, Claude Shannon demonstrated in 1937 how digital circuits could implement Boolean logical functions, now fundamental to modern processor design [Shannon (1938)]. The creation of the transistor in 1947 [Brinkman et al. (1997)], its subsequent implementation in semiconducting silicon and the era of the integrated circuit led the way in miniaturization and associated speed increases. The use of silicon microprocessors is another typical feature of conventional computing, and underlies almost all modern computers facilitating many aspects of life and scientific development.

Von Neumann architecture is the classical architecture of silicon microprocessors and conventional computers and consists of several interconnected units and buses allowing communication between them. A control unit coordinates the operation of other units by fetching instructions from the program, consigning execution to the arithmetic logic unit and writing results back to memory. The control unit also governs reading and writing to input/output devices and also external storage devices. An arithmetic logic unit performs mathematical and logical functions as designated by the control unit. Memory stores both the program and the data, and is referenced by the control unit and arithmetic logic unit. This implementation of a computer is used today, and represents one of the primary features of conventional computing. It also represented the first practical implementation of a computing machinery similar (in that is contains subsystems operating on stored programs with input and output) to Turing's universal machine.

The use of silicon microprocessors in all conventional computing devices gives the conventional paradigm several distinctive traits. Microprocessors are electronic, and require a constant source of power to operate. They are powerful, able to compute and store vast amounts of data, though waste electricity causes the generation of heat which must be removed before it damages the chip. Computers are robust when correctly isolated from their environment and though they cannot directly interact with their surroundings, they can control peripherals and accessories which can. As speed is limited by transmission delays between transistors, chip fabrication techniques need to improve to resolve increasingly smaller feature sizes, but we may be reaching the limits of this approach [Zhirnov et al. (2003)] [Keyes (2001)]. The complexity of their construction means computers are often costly to build and to design, but economies of scale and a massive global demand keeps prices at a reasonable level. Processors execute some operations in parallel, but most programs are designed to operate in a serial manner. Some parallel problem solving exists, but typically only on a small scale.

## 2.2. Unconventional Computing

Unconventional computing, by comparison, seeks new ways to solve problems that are not bound by conventional computing architectures. Unconventional computing is a very wide field, encompassing a vast range of information processing paradigms. Unconventional computing tends not to compete with conventional computing's strengths, but rather to explore other types of computation, new materials and crucially, new settings in which computers can operate.

That said, some methods do seek to improve the rate of computation [Lin et al. (2010)], or to solve problems a conventional computer cannot [Potgieter (2006)]. Others to reduce energy consumption [Landauer (1961)] [Vitányi (2005)], or do away with electrical signals altogether [Tominaga et al. (2001)]. Some are self-healing [Ramachandran et al. (2012)], or able to continue functioning despite damage [Abelson et al. (2000)]. Some are made of biological components [Adleman (1994)], and others are repurposed life [Adamatzky (2007)].

One of the simplest forms of unconventional computing is to implement a paradigm that operates on a different form of logic to Boolean. Kleene's three-

valued logic utilises three possible values rather than Boolean's two; *False*, *True* and *Unknown* [Kleene (1952)] [Malinowski (2007)]. 3-Valued Łukasiewicz logic uses 0, 1 and ½ [Lukasiewicz and Borkowski (1970)] [Hájek (1998)] and can be extended to any arbitrary number of logic values including infinite, [Hay (1963)] similarly to fuzzy logic [Zadeh (1965)] [Perfilieva and Mockor (1999)]. Logic may also be reversible, where every output has a unique input [Toffoli (1980)] [Fredkin and Toffoli (2002)] [Vitányi (2005)]. As reversible computing generates less physical entropy, heat generation is lower [Bennett (1982)]. Numerical representation may also vary from binary, including balanced ternary (-1, 0 and 1) [Frieder (1973)] and real computers [Blum et al. (1988)] [Blum (1998)].

Quantum computing leverages the ability for particles to exist in superposition, such that a quantum bit (a qubit) contains a superposition of both 0 and 1. Quantum algorithms such as Shor's algorithm for integer factorisation [Shor (1994)] have generated huge interest in quantum computation, with some practical implementations beginning to appear [DiCarlo et al. (2009)] [Johnson et al. (2011)].

Cellular automata are an example of a radically different computing architecture consisting of discrete cells in a matrix with a set of possible states, and transformation rules that govern how each cell evolves over time. The simplest examples are the Elementary Cellular Automata (ECAs); one dimensional arrays of cells with two possible states which evolve each time step by comparing the state of the cell and its two immediate neighbours with a state transition function [Wolfram (2002)]. Even this simple model of computation is universal [Cook (2004)]. Wolfram classes detail the complexity of behaviour of ECAs. There are four classes:

*Class 1*: Almost all initial states evolve into a uniform final state or alternates between two states.
*Class 2:* Many uniform or repetitive final states exist, but they exist of repeating simple structures.
*Class 3:* The ECA evolves in an apparently random manner.
*Class 4*: Localised structures form and interact in 'interesting' ways.

Famous two dimensional examples include von-Neumann's universal constructors [von Neumann (1966)] and Conway's Game of Life [Gardner (1970)], a

biologically inspired game that has been shown to be universal [Berlekamp et al. (1982)].

Another biologically inspired computing paradigm are P-Systems, also known as membrane computers [Paun (1998)] [Păun (2006)]. Drawing inspiration from biological modelling techniques, a P-System consists of a hierarchy of membranes each containing symbols and rules that govern the evolution of those symbols. The idea of a computer able to divide and grow in response to the complexity of a problem has generated interest [Paun (2001)] as P-systems can, in principle, solve NP-Complete problems in reasonable time[2].

It should be noted that at present, all the computing paradigms mentioned in this article can be simulated on a conventional computer (itself a universal Turing machine). Though they could in theory offer advantages in speed of execution or increased integration with the physical world, they are no more powerful than a Turing machine once implemented in physical reality. This is the physical variation of the Church-Turing thesis; that all physically computable functions are Turing computable. [Cotogno (2003)]

***Church-Turing Thesis**: Any function that is intuitively computable is computable by some Turing machine (Turing-computable for short). [Piccinini (2011)]*

***Physical Church-Turing Thesis**: Any function that is physically computable is Turing computable. [Piccinini (2011)]*

## 2.3. Unconventional Molecular Computing

Molecular computing is the use of populations of molecules to implement computational devices. Molecular computing has been heavily researched, with a variety of molecules and supramolecular assemblies [Lehn (1988)] [Katz (2012)] showing promise for implementing logical functions. For example, crown-ethers with ions as inputs will form gels or remain fluid, mimicking many common logic functions [Qi et al. (2012)]. Macrocycle-clip complexes with ions as inputs change colour, implementing a 2-input NOR gate. [Cheng et al. (2005)].

[2] Implementing this would, however, be impossible.

A common benchmark for molecular computing is the creation of half-adders. One of the simplest logical circuits but also one of the most important; the half-adder adds two one-bit numbers together. As half-adders require XOR gates and AND gates, there are many examples of XOR and AND implementations. [Prasanna de Silva and McClenaghan (2000)] uses hydrogen and calcium ions, and the 'output' is optical transmittance. [de Silva et al. (1997)] shows an anthracene fluorophore (a fluorescent molecule) with amine and crown ether moieties (functional groups attached to a molecule), where the fluorescence is quenched unless hydrogen and sodium ions are present, implementing an AND gate.

Molecular computing suffers from low execution speeds, taking 0.1 ms for an enzyme interaction compared to just nanoseconds for a CPU operation [Conrad (1985)]. One possible way to mitigate this is via molecular electronics, using molecules [Aviram and Ratner (1974)] [Reed et al. (1997)] [Flood et al. (2004)] and supramolecular assemblies [Schenning et al. (2004)] to implement electrical components.

The use of biological molecules for molecular computing implementations is common. Of the biological molecules, DNA is one of the most heavily researched [Amos (2005)]. Perhaps the most famous example of DNA computing is Adleman's paper on solving combinatorial problems [Adleman (1994)]. By using the massively parallel nature of molecular interactions, the nature of DNA base bonding, and coupled with the ability to create custom DNA strands a solution to a seven-point directed Hamiltonian path problem was found.

Figure 2.2 a) shows the example used by Adleman. The directed Hamiltonian path problem involves proving the existence of a path that starts at a specified vertex ($v_{in}$), ends at a specified vertex ($v_{out}$) and which visits all vertices along the way once by following directed edges. By encoding all the vertices in the graph with 20-long chains of DNA bases, and all the edges with 20-long chains of bases such that the first half of the edge strand would bind to the corresponding origin vertex strand, and the second half to the destination vertex strand, all possible paths through the system could be rapidly produced.

The rapid generation of a huge number of random paths through graph represents the first step of the algorithm. The remaining steps involve filtering invalid paths out of the sample:

i. Generate a large number of random paths through the graph that can be reasonably expected to contain a valid solution if one exists.
ii. Remove all paths that do not begin with $v_{in}$, and do not end with $v_{out}$.
iii. Keep only paths of length |V| where V is the number of vertices in the graph.
iv. Keep only those paths which visit all vertices once.
v. If any path remains, return *yes*. Else *no*.

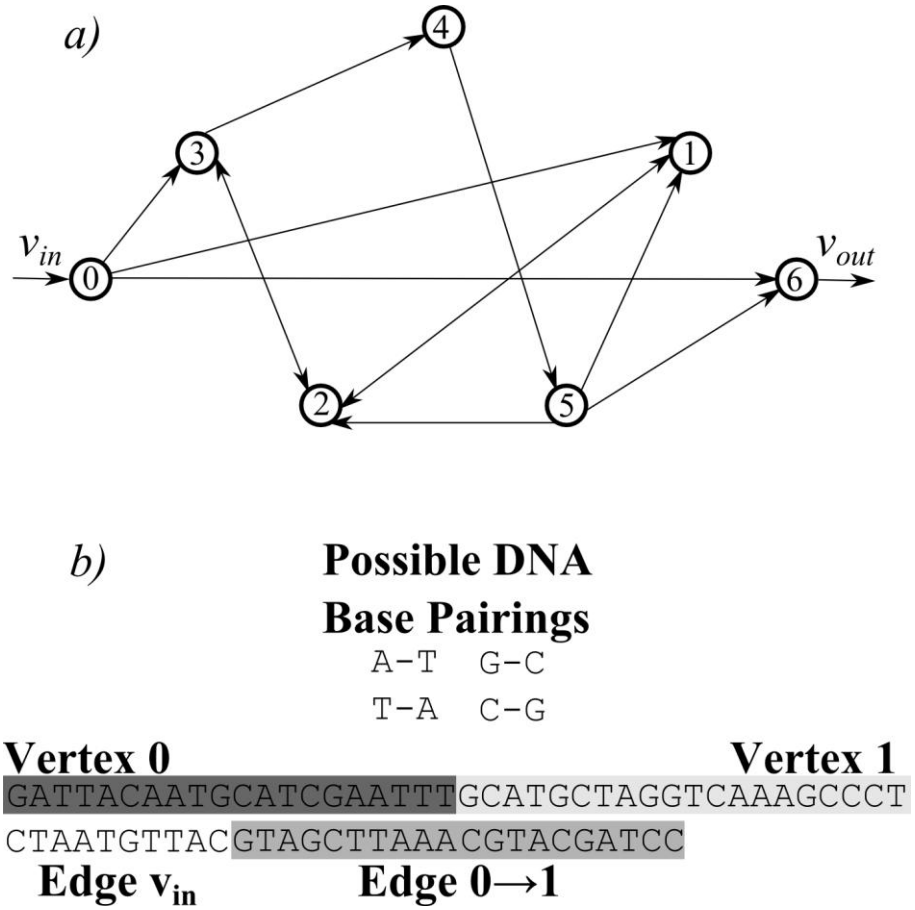



*Figure 2.2: Adleman's massively parallel DNA computer. a) The directed 7-node Hamiltonian path problem, as solved by Adleman's DNA computer. The solution to this problem is simply 0→1, 1→2, 2→3, 3→4, 4→5, 5→6, though Adleman's computer only proves the existence of this path, and does not output the path itself. b) DNA bases only bind in specific pairs. Example DNA strands showing how two vertices can be joined by an edge, and how a possible solution is started and terminated.*

Another possible application of DNA is in self-assembling systems. Self assembly is the process of a disordered system that forms order via the local interactions of components without any overarching plan. DNA can self assemble into a variety of structures [Li et al. (2009)] and direct the assembly of proteins [Niemeyer (2002)]. Wang tiles [Wang (1960)] are an example of a self-assembly computing logic and DNA [Winfree et al. (1998)] 'tiles' could implement these. Phospholipids self assemble to form lipid bilayers and vesicles, intrinsic components of biological life.

A related form of unconventional computing is chemical computing. Chemical computation is the use of chemical samples and reactions between them to compute solutions to problems. One of the most common chemical reactions used is the Belousov-Zhabotinsky Reaction (BZ Reaction) [Belousov (1959)] [Ichino et al. (2008)], a mixture of bromine and acid which oscillates between colourless and coloured as the concentration of cerium ions varies. These oscillations have been used in collision based computation [Adamatzky (2004)] producing logic gates and circuits as waves collide and fragment [Adamatzky et al. (2012)].

One possible approach is to implement conventional von-Neumann architecture on a non-conventional substrate (i.e. not silicon). Graphene (carbon monolayers) shows promise in implementing transistors with higher switching rates [Lin et al. (2010)] and computers with improved battery capacities [Stankovich et al. (2006)]. Conductive polymers could lead to flexible computers [Shirakawa et al. (1977)] [Inzelt (2012)].

## 2.4. Molecular Switches

Molecular switches are molecules with multiple stable forms which can be switched between these forms reversibly via application of a stimulus, such as light [Exelby and Grinter (1965)], temperature [Mehta and Sen (2009)], the presence of other chemical species [Qi et al. (2012)] or pH [Pasparakis et al. (2009)]. In some cases, this may be a change in the molecule's shape or alignment. In other cases, it could be the locking/unlocking action of two mechanically interlinked molecules in a supramolecular construct [Green et al. (2007)]. For example, molecular shuttles are a self-assembled supramolecular construct consisting of a movable macrocycle 'bead' on a polyether 'thread' transitioning between hydroquinol 'stations', with

triisopropylsilyl ‘stoppers’ at each end to prevent the bead falling off [Anelli et al. (1991)]. It could also be the capture/release of two molecules with a host/guest relationship [Shinkai et al. (1981)] [Harris et al. (2011)]. The different forms of the molecules have different properties and behaviours.

Molecular switches play an important role in natural biology, such as switching proteins in ion channels and enzymes. Ion channels permit or restrict the passage of ions via conformal change in response to membrane potential or ligands [Hille (2001)]. Enzymes can alter rates of chemical reactions, and can undergo conformal changes in response to chemical signals [Hammes (2002)]. Research into synthetic switches that control gene expression is an important aspect of synthetic biology [Weber and Fussenegger (2011)].

## 2.5. Photochromic Molecular Switches

Photochromic molecules [Exelby and Grinter (1965)] [Dürr and Bouas-Laurent (2003)] are a sub-type of molecular switch with multiple stable forms which can be reversibly switched between these forms via the absorption of electromagnetic radiation within an absorption spectrum. The most common application familiar to most is in optically reactive spectacle lenses [Le Naour-Sene (1981)], which use spiropyrans or metal dithizonate in a polymer matrix to switch from a transparent to an opaque form upon absorption of ultraviolet light. There are many families of photochromic molecules [Guglielmetti (2003)] including some photochromic supramolecular constructs [Vögtle et al. (1993)] and some change their chiralty in response to light [Feringa et al. (2000)]. Some photochromic molecules change colour, others change between a fluorescent and non-fluorescent form.

There are several issues to consider when working with photochromic molecules. The first issue is that not all photons incident to a sample are absorbed by the photochromic molecules. The molar absorptivity of the photochromic molecule ($\varepsilon$, how strongly a molecule absorbs photons) and the thickness of the sample determine the proportion of photons which are absorbed. Remaining photons simply pass straight through the sample.

The second is the quantum yields ($\Phi$). The quantum yield of a transition is the probability that a photon which was absorbed will induce an event (such as a

form change). The quantum yield is dependent on the wavelength of absorbed photons and the current molecular form. A two-state photochromic molecule with forms A and B will hence have quantum yields, $\Phi^{\lambda}_{A\to B}$ and $\Phi^{\lambda}_{B\to A}$. $\Phi_{A\to B}$ and $\Phi_{B\to A}$ will have peak wavelengths for which the quantum yields are high, with decreasing quantum yields away from that point. A quantum yield is a dimensionless value between 0 (no absorbed photons cause an event) and 1 (all absorbed photons cause an event). [Valeur (2001)]

A third issue is thermal transitions. Though energy imparted by photons is the primary means of form transitions in photochromic molecules, thermal energy can also cause transitions. For any given temperature, a sample of molecules will exist in a Boltzmann distribution of kinetic energy; some have low energy, and some have a very high energy. Above a threshold of kinetic energy (the activation energy), the molecule will spontaneously change form. Typically the non-fluorescent form is the most thermodynamically stable, so random thermal transitions will trend towards the non-fluorescent form, as the transition from fluorescent form to non-fluorescent form requires less energy than the opposite. However, as all transitions can occur via thermal effects, a photochromic molecular sample left at a constant temperature will 'thermally relax' towards an equilibrium in which the proportion of molecules in each form are dependent on the temperature and the properties of the molecule. [Guglielmetti (2003)] A summary of these issues can be seen in *Figure 2.3*.

A variety of reversibly switchable photochromic molecular probes exist. Some have additional interesting properties. Spiroamido-rhodamine [Kolmakov et al. (2010)] switches from a non-fluorescent form to a fluorescent form on absorption of ultraviolet light, and rapidly relaxes thermally. This rate of relaxation requires a high intensity of ultraviolet light to convert and the fluorescent form is highly unstable. Diheteroarylethenes can be used for photochromic Förster resonance energy transfer (pcFRET) imaging [Giordano et al. (2002)], where the form of the diheteroarylethene molecule governs if FRET occurs (See section 2.6: Fluorescence for more details). Azobenzene can be bound to a protein and allow allosteric control (allowing/inhibiting the binding with other proteins) depending on the form of the azobenzene molecule [Volgraf et al. (2006)]. Mutated Dronpa proteins can also be switched [Ando et al. (2007)] to improve contrast in biological imaging applications.

Photochromic molecules offer possibilities to overcome the difficulties of inter-component signalling in molecular logic circuits. [Raymo and Giordani (2002)] uses three fluorescent molecules in a container, where the emission wavelengths of these molecules are tailored to the absorption wavelength of a spiropyran's three forms. The spiropyran is contained in a separate cell. The result is that specific wavelengths of fluorescence are absorbed as they pass through the spiropyran cell depending on the form of the spiropyran. Though it allows for permissive communication via allowed wavelengths, it requires careful positioning of the two cells, uses three additional optical inputs and has low efficiency.

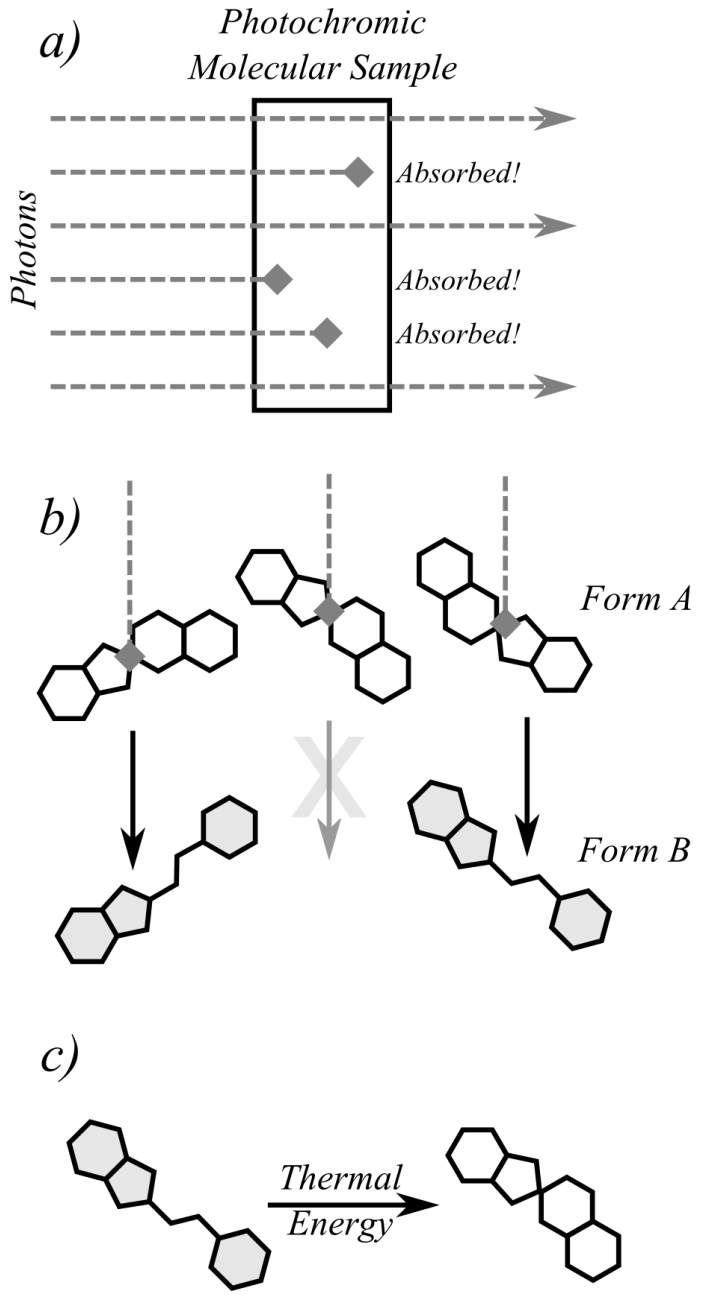



*Figure 2.3: Three issues with photochromic molecules. a) Not all photons incident to a sample will be absorbed, many will pass straight through. b) Not all absorbed photons cause an event such as a form change. This is given by the quantum yield. c) Thermal energy can also cause events, trending to cause samples to thermally relax to their most stable form.*

## 2.6. Fluorescence

A common property in photochromic molecules is that one form is fluorescent. Luminescence a term describing the general phenomena of the emission of photons from an electronically excited molecular species. Fluorescence describes

the special case when the excitation is due to the absorption of photons (compared to thermoluminescence for heat, or bioluminescence for a biochemical process), and the emitted photon is produced due to immediate relaxation of the excited molecule (compared to phosphorescence in which there is a delay, as leveraged by ‘glow in the dark’ products). [Valeur (2001)] A fluorescent molecule is known as a fluorophore.

The absorption of photons of appropriate wavelengths (dependant on the molecule) causes orbital electrons to be excited to higher energy levels. Fluorophores will have excitation spectra; absorption bands of wavelengths that will cause excitation. These spectra have a peak absorption wavelength, with decreasing rates of absorption further from the peak, and each wavelength of photon will have an associated quantum yield of fluorescence. [Valeur (2001)]

The energy lost when the electron relaxes is released as a photon. This will also have an emission spectra with a peak wavelength, though due to energy lost during the emission process the emitted photon will be of a lower energy (and hence longer wavelength) than the exciting photon. This energy loss is known as Stoke’s shift, as shown in *Figure 2.4*. [Valeur (2001)]

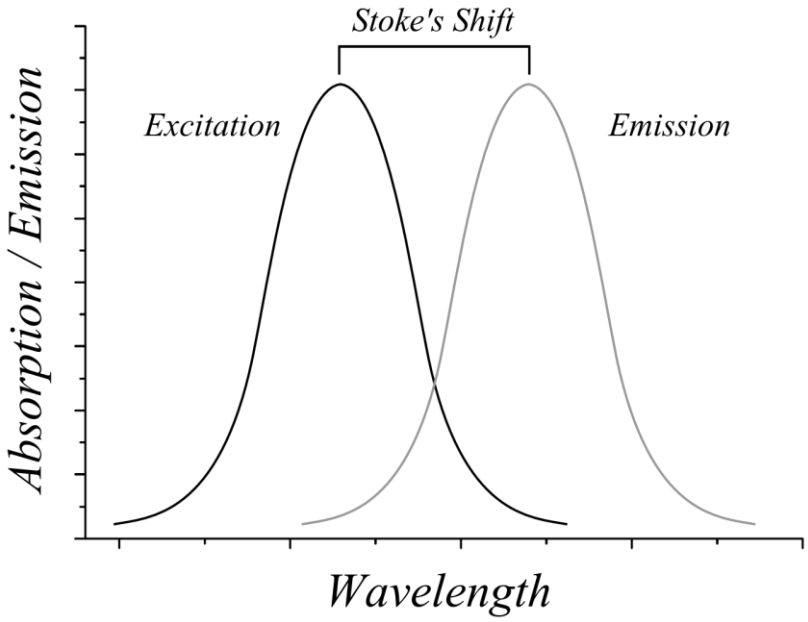



*Figure 2.4: Stoke's shift is the loss of energy during the fluorescence process. Emitted photons will have lower energy (and hence longer wavelength) than the excitatory photons.*

An issue with fluorophores is photobleaching. Photobleaching is the photochemical damage of a fluorophore by excitatory photons until it no longer functions. In photochromic molecules, this effectively fixes it into the non-fluorescent form, reducing the fluorescent dynamic range of a photochromic molecular sample. The exact properties of the bleached state vary between species of photochromic molecule. The rate of photobleaching can be expressed as a quantum yield $\Phi_{bleach}$ which describes the probability of an excited molecule becoming bleached.

One use of photochromic molecules is in the imaging of proteins within living systems. By using photochromic optical probes targeted for specific proteins, excitatory stimulus causes the probes to fluoresce depending on their form, which can then be detected. The areas of fluorescence then correspond to the concentration of protein. The use of fluorophores can be problematic as excitation will cause all fluorophores to fluoresce - including endogenous fluorophores - both inside and outside the focal plane [Yan et al. (2011)], resulting in low contrast. The irreversible uncaging of caged fluorophores can be used [Mitchison (1989)] to alleviate this -

only fluorescing once uncaged with ultraviolet light - but the fluorophores cannot be recaged. By using switching fluorescent probes, the fluorescence signal before switching the probes to the fluorescent form can be subtracted from the signal after switching to the fluorescence form, improving contrast dramatically.

Förster resonance energy transfer (FRET) is a useful phenomenon leveraged in biological imaging. Two molecules (dubbed the donor and acceptor) can transfer energy without intermediate fluorescence if the emission spectrum of the donor overlaps the absorption spectrum of the acceptor. [Valeur (2001)] This energy transfer only occurs when the donor and acceptor are in very close proximity, providing a means of determining the distance between two molecules. In biological imaging the donor and acceptor molecules are tagged to other biological molecules (i.e. proteins), and excitatory stimulus will cause the donor to fluorescence if the molecules are not proximate, and the acceptor to fluoresce with a different wavelength to the donor if they are (i.e. during protein interactions).

## 2.7. Spiropyran Photochromic Molecular Switches and NitroBIPS

The species of photochromic molecule studied in this thesis are the spiropyrans. The photochromic properties of spiropyrans were first discovered in 1952 [Fischer and Hirshberg (1952)] and potential applications to data storage were rapidly identified. [Hirshberg (1956)]. NitroBIPS (1',3'-dihydro-1',3',3'-trimethyl-6-nitrospiro[2H-1-benzopyran-2-2'-2H-indole], nitrobenzospiropyran or 6-nitro-BIPS and shortened to NBIPS) is one such spiropyran.

NitroBIPS is a leuco dye, with a colourless non-fluorescent spiropyran form (SP), and an intense purple fluorescent merocyanine form (MC). The SP → MC transition is hence known as 'colouration', and the reverse as 'decolouration'. Colouration occurs upon absorption of an ultraviolet photon, with an absorption peak ~365nm ($\lambda_c$) [Marriott et al. (2008)] with an associated quantum yield $\Phi_c$. Decolouration occurs upon absorption of a visible wavelength photon, with an absorption peak ~543nm green ($\lambda_d$) with an associated quantum yield $\Phi_d$. The absorption of green light is the cause of MC-form molecules' purple colour. MC molecules are fluorescent via an excited MC* state with an excitation wavelength equal to $\lambda_d$ and a quantum yield $\Phi_f$. Emission is in the orange range. ($\lambda_f$ ~630 nm). A summary of these transitions can be seen in Figure 2.5. The ability of NitroBIPS to

fluoresce only in the MC form allows us to measure the proportion of forms in a sample of NitroBIPS. Irradiating an MC-form molecule with visible light either causes fluorescence, or decolouration. For populations of molecules, a sample of MC-form molecules fluoresces and decolours simultaneously.

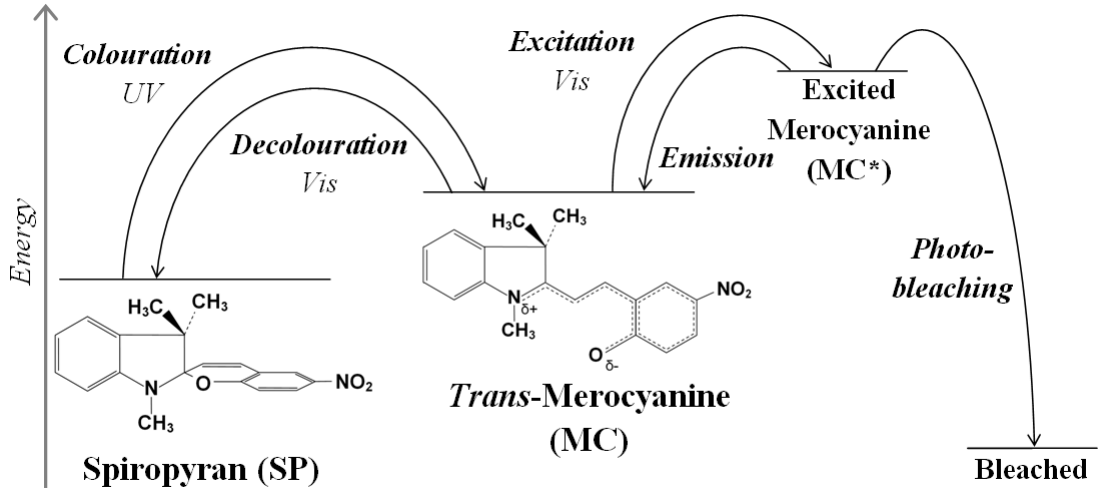



*Figure 2.5: Forms of NitroBIPS and their relative energy levels, and the transitions between them. Each transition is named and labelled with the stimulus required to induce the change.*

The absorption peaks $\lambda_c$ and $\lambda_d$ are approximate, as NitroBIPS is solvatochromic; the less polar the solvent, the higher the molecular mobility and hence the less energy is required for form transitions, and hence the absorption peak shifts to a longer wavelength [Wojtyk et al. (2000)] [Görner and Matter (2001)]. For similar reasons quantum yields improve in lower polarity solvents, as high as 0.75 for $\Phi_d$ in toluene [Görner and Matter (2001)]. Though this property would lower the use of NitroBIPS in biological settings (being typically highly polar), quantum yields improve when NitroBIPS is bound to a protein or membrane [Yan et al. (2011)]. Additionally, the peak wavelength of emitted photons and the peak wavelength of excitation vary depending on the wavelength used to colourise the sample; different ultraviolet wavelengths cause different Merocyanine isomers to be formed, each with different absorption and emission spectra [Wohl and Kuciauskas (2005)].

Like other photochromic molecules, NitroBIPS is subject to photobleaching and thermally induced transitions. As SP is the more stable, a sample of NitroBIPS will relax to a majority-SP equilibrium that is dependent on the temperature. This

effect limits the ability for spiropyrans to be used as long term data storage [Wojtyk et al. (2000)]. The rate of thermal decay is solvent dependant, with lower polarities decaying quicker due to less energy being required for transitions. Reported rate constants at for nitro-substituted spiropyrans suggest that NitroBIPS undergoes the most rapid thermal relaxation compared to other similar spiropyrans [Wojtyk et al. (2000)]. Photobleaching occurs when the excited MC* molecule undergoes inter-system crossing in which it irreversibly enters a non-fluorescent state (referred to as B) and ceases to function.

## 2.8. Systems and Synthetic Biology

Systems biology is the study of the regulatory networks and molecular interactions in biological systems to produce models and analyses. [Kitano (2001)] These methods allow us a window into the underlying processes that govern life and how we might modify these processes. The engineering of existing biological entities to implement new behaviour not seen in nature is known as synthetic biology, and is an extension of systems biology that bridges the gap from understanding to manipulation.

Synthetic biology has two primary and competing methodologies: top-down and bottom-up. Top-down synthetic biology is taking existing biology and editing it. By taking a systems biology approach to life and understanding the processes that govern it, it can be altered. Taking entire blocks of genetic code from one life form and introducing it into another is one method. For example, introducing Green Fluorescent Protein (GFP) genes into the same regulatory sequence as another protein, allowing protein production to be monitored via fluorescence [Chalfie et al. (1994)]. Colonies of engineered *Escherichia-coli* have been shown to exhibit the functionality of logic gates, with multiple colonies communicating to form logic circuits [Tamsir et al. (2011)]. However, the production of binary logic gates in cells by introducing new transcription regulatory networks is difficult, as existing transcription networks interfere [Silva-Rocha and de Lorenzo (2008)]. The creation of a minimal cell – the simplest possible organism – is an important aim of synthetic biology as it would allow for the introduction of new processes without first understanding all existing processes. [Heinemann and Panke (2006)] [Forster and Church (2006)] [Gil et al. (2004)].

The alternative is bottom-up synthetic biology, which begins from simpler chemical elements and seeks to develop reusable modules from which life can be synthesized. The production of an artificial cell (referred to as protocells, chemical cells or ‘chells’) capable of reproduction and reacting to the environment would be the first step to an extendable framework where additional modules and routines could be introduced. There is no accepted measure by which a chell could be declared ‘alive’ [Cronin et al. (2006)], but production of bespoke life is a major goal of synthetic biology.

What components are necessary for a minimal chell is generally clearly agreed upon [Porcar et al. (2011)] [Moya et al. (2009)]; a container for internal mechanisms and protection (i.e. a cell’s membrane), some internal functional elements that execute the chell’s self regulation and environmental interactions (i.e. a cell’s metabolism), and replicable data storage so the chell can reproduce (i.e. DNA). How exactly each of these components could be implemented is less clear.

One possible way of regulating chell processes is via molecular switches, either as internal mechanisms or as a way for the environment to interact and modify chell behaviour. Though this thesis does not implement any synthetic biology, the choice of photochromic molecules for implementing universal computation was made with an eye towards possible synthetic biology applications both from the top-down and the bottom-up. NitroBIPS and spiropyrans in general are strategic assets in the pursuit of methods for embedding ‘interfaceable’ computation in cells and chells.

NitroBIPS and spiropyrans in general are of interest to synthetic biology due to the numerous possibilities for interaction with biological processes. NitroBIPS and other spiropyrans can be bound to proteins [Sakata et al. (2005)] and enzymes [Aizawa et al. (1977)] via the introduction of a linked probe, and can control the dipolar interactions of the targeted molecule depending on the state of the spiropyran. Spiropyrans bound to single chain lipids can form small unilamellar vesicles (SUV) where the interior content of the SUV will be released if the spiropyrans are in the MC form. [Ohya et al. (1998)] The MC-form spiropyrans cause the vesicle membrane to be perturbed, allowing molecules to escape without permanent damage to the membrane; the higher the proportion of MC-form molecules, the greater the rate of release. Channel proteins embedded in membranes

can also be controlled via attached spiropyrans [Koçer et al. (2005)]. Spiropyrans are encapsulable in vesicles 200 to 500nm in diameter [Carol A. Jennings et al. (1997)]. Spiropyrans can also be reversibly intercalated to targeted DNA strands [Andersson et al. (2008)] without prior modification to the DNA molecule. This potentially allows for the selective reversible inhibition of parts of DNA strands by introducing spiropyrans. Some potential applications of the results in this thesis to synthetic biology are further discussed in *Chapter 9: Discussion and Conclusions*.

# Chapter 3: Computational Model and Programmability

A basic description of photochromic molecular behaviour is given in *2.5 Photochromic Molecular Switches* and *2.7 Spiropyran Photochromic Molecular Switches and NitroBIPS.* From this behaviour, we must form a basic machine model which allows for the storage of data and the execution of logic functions, which can then be experimentally implemented. This section also details how to program photochromic systems, converting from a logic circuit to a series of light pulses and fluorescence measurements.

## 3.1. Computational Model

### *3.1.1 Registers*

A register can be defined with three basic actions; increment, decrement and read. There are two fundamental actions in a photochromic memory system; colourise and decolourise, representing the application of colourisation-wavelength and decolourisation-wavelength photons. Colourise is equivalent to incrementation, and decolourisation fulfils both decrementation and reading. These functions are applied to a sample of photochromic molecules, altering a measurable property; in this case the proportion of MC-form molecules which can be measured via the intensity of fluorescence under excitation stimulus.

The computational model here could be applied to any photochromic molecule or indeed molecular switch, with two stimuli that increment and decrement

an observable property. For NitroBIPS, decolouration and measurement are the same action as the photon wavelengths required to decolour and illicit fluorescence are the same. For other molecules, the decolour and measure actions may be separate.

***Writing***

As described in 2.7, colouration and decolouration are the two stimuli that cause a form-change in NitroBIPS. For NitroBIPS, ultraviolet-wavelength photons colourise, and visible-wavelength photons decolourise. Colourisation causes an increase in the proportion MC-form molecules in the NitroBIPS sample, and decolouration causes a decrease.

Abstracting away from molecular forms and wavelengths, instead consider that a sample of photochromic molecules can store a number of distinguishable values. A sample of molecular switches can store data as the relative concentrations of molecules in each of the two stable states, giving a non-binary integer register. An integer register stores a value, from $V_0$ to $V_{max}$, where max is the maximum capacity of the register. For NitroBIPS, we designate an increase in the proportion of MC-form molecules (and hence an increase in measured fluorescence) as an increase in the value of the sample.

The two fundamental actions colourise and decolourise hence serve to increment and decrement the register's value. The duration of pulses is application, hardware, sample and molecular species-dependant, but to abstract from exact durations and indeed exact stimulus type, we can represent colouration as $\uparrow_y$ and decolouration as $\downarrow_y$, where $y$ is the change in the value caused by the pulse. i.e.:

| | | |
|---|---|---|
| $V_x + \uparrow_y = V_{x+y}$ | $x + y < max$ | Equation 1 |
| $V_x + \uparrow_y = V_{max}$ | $x + y \geq max$ | Equation 2 |
| $V_x + \downarrow_y = V_{x-y}$ | $x > y$ | Equation 3 |
| $V_x + \downarrow_y = V_0$ | $x \leq y$ | Equation 4 |

where Equation 2 is the ceiling restriction and Equation 4 is the floor restriction. These restrictions are crucial, as they represent the mechanism by which photochromic memory can execute logic functions.

***Reading***

Reading a photochromic register requires us to determine the proportion of MC-form molecules in the sample. MC-form molecules fluoresce under excitation stimulus, so the higher the intensity of recorded fluorescence under excitation stimulus, the higher the proportion of MC-form molecules.

Rather than have a separate read action, photochromic molecules fluoresce under decolouration stimulus. Hence, decrementation and reading a register occur simultaneously, as shown in Figure 3.1. The value of a register can be determined any time a decrement action is enacted, regardless of the amount of decrementation. This does mean a register cannot be read without some amount of decrementation, and that a register's value will necessarily be changed by a read action, i.e. reading is destructive.

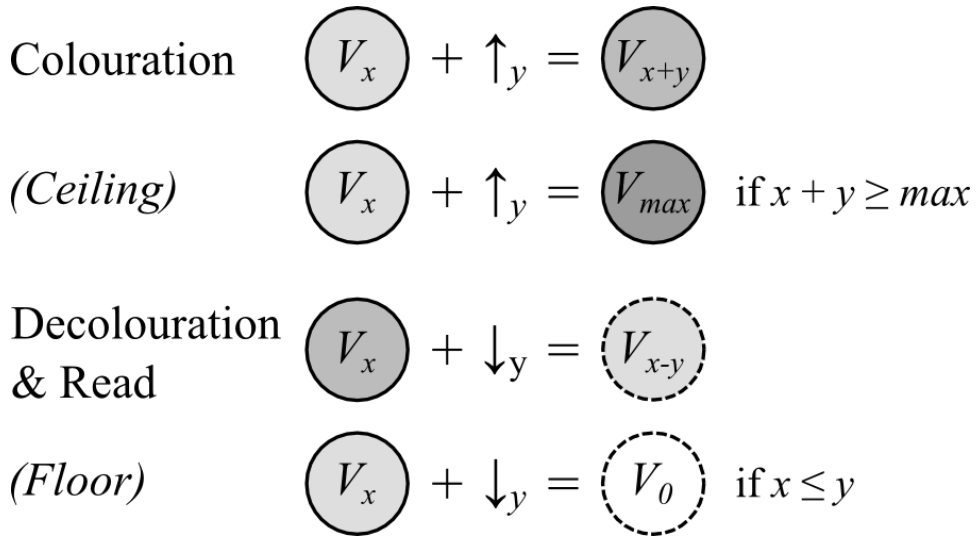



*Figure 3.1: The two basic actions of a photochromic register, and two additional edge cases representing the floor and ceiling restriction, corresponding to Equation 1 to Equation 4. Note that decolouration and read are the same action. The dashed outlines represent the ability to determine the value of* x *when this action is enacted.*

If a register is to be read, but kept at its previous value, it could (for example) be subjected to $\downarrow_1$ followed by $\uparrow_1$ stimulus, providing a decrement action to allow for reading while returning the register to its previous state. The exception to this case would be for a register already in $V_0$, where no colouration stimulus would be required.

### *3.1.2 Logic Gates*

A logic gate is an extension to our model of registers. It consists of a sequence of pulses which enact the gate inputs. A gate begins by being initialised to $V_0$ and the output of the gate is the value the gate ends in, which is then read like a register. The logic gate is subject to an input-dependant sequence of light pulses, leaving the gate in the output value. Some pulses require tagging to show their relation with the inputs to the gate.

> *{n, ↕$_y$}:* A stimulus that either occurs or does not depending on the inputs to the logic gate. A gate's inputs are labelled *a, b, c...* and for each execution of the gate they will each be *True* or *False*. For example, a gate may have an input *a* which has an associated pulse *{a, ↑$_1$}*. If *a* is *True*, this is equivalent to a ↑$_1$ pulse. If *a* is *False*, this is equivalent to ↑$_0$ or no pulse at all.

An *n*-arity logic gate is defined as [*c, (p*), o*] where:

> *c*: *c* is the capacity of register required to operate. Photochromic logic gates are extensions to photochromic registers, and require certain range of values to operate. Most gates require a minimum capacity, rendered as *c+*. Some require an exact capacity to operate, rendered as *c!*.
>
> *(p*):* An ordered list of pulses. It contains a combination of tagged pulses to enact the gate inputs, and 'moderator' pulses which occur regardless of the pattern of inputs. Not all gate inputs need corresponding input pulses; some logical functions are not dependant on all their inputs.
>
> *o*: The output pulse. This will always be a ↓ pulse with duration equal to the highest possible value the gate can be in at the end of *(p*)*. The output pulse allows the determination of the final state of the logic gate as a read pulse does with a register, and resets the gate to $V_0$ for subsequent executions. For most logic gates, a reading of $V_0$ implies *False*, and $\geq V_1$ implies *True*. This is a '*unique*' gate; any fluorescence during the output phase (above background noise) implies truth.

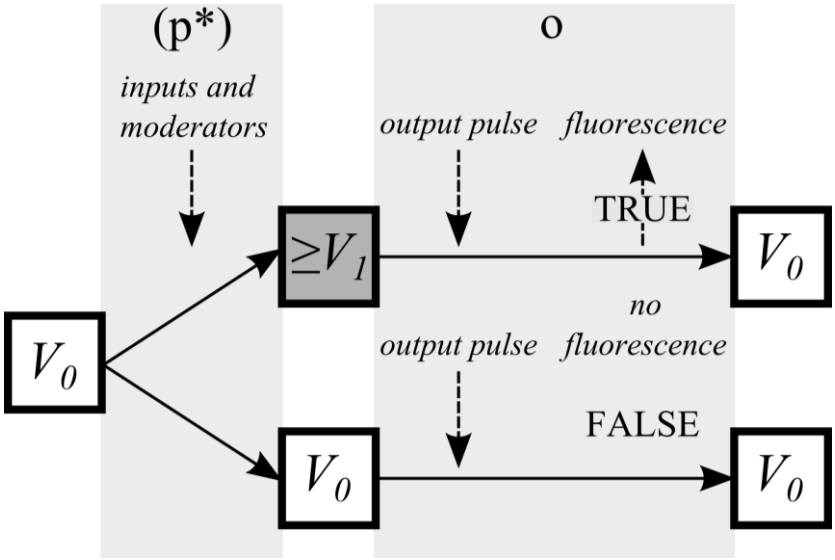



*Figure 3.2: Logic Gate operation for unique logic gates. Depending on the inputs to the gate, the gate can be left in* $V_0$*, or a higher value* $\geq V_1$*. The output pulse will then determine the state of the logic gate via the level of fluorescence detected. Note any transition from a higher value to a lower value will cause fluorescence. Only fluorescence during the output pulse must be considered.*

A summary of logic gate operation in this definition is shown in Figure 3.2. An example in this definition is the 2-input AND gate:

$$2\text{-AND} = [3+, (\{a, \uparrow_1\}, \{b, \uparrow_1\}, \{\downarrow_1\}), \downarrow_1]$$

We see that an AND gate requires 3 values, $V_0$, $V_1$ and $V_2$. A moderator decolours by one value before the output is measured, so only if both inputs were true and set the gate to $V_2$ will the output be positive, as summarised in Table 1.

| **Input *a* Value** | **Input *b* Value** | **Input *a* Pulse Value Change** | **Input *b* Pulse Value Change** | **Moderator Pulse Value Change** | **Final Value Before Output** | **Output Pulse Value Change** | **Truth Value** |
|---|---|---|---|---|---|---|---|
| F | F | 0 | 0 | -1* | 0 | -1* | F |
| **T** | F | +1 | 0 | -1 | 0 | -1* | F |
| F | **T** | 0 | +1 | -1 | 0 | -1* | F |
| **T** | **T** | +1 | +1 | -1 | 1 | -1 | **T** |

*Table 1: Possible execution paths for a 2-AND gate. Negative value changes marked with an asterisk (*) do not affect the system as the register was already in* $V_0$ *and those changes were subject to the floor restriction.*

eNBIPS gates have three properties not common to traditional logic gates. The first is *uniqueness*. A gate is unique if the output is *False* for $V_0$ and *True* for $\geq V_1$. This can be imagined as the gate being true if there is any fluorescence (above a threshold for background noise) during the output pulse. A gate is *non-unique* if the output is *True* for a given value, and *False* for all others. For example, the non-unique gate XOR is *True* for $V_1$, and *False* for $V_0$ and $\geq V_2$. Uniqueness is a desirable trait as it simplifies implementation by not relying on a specific value, but rather any level of fluorescence above the lower bound for $V_1$.

The second property is *equality of output*. As stated for uniqueness, a unique gate is one with any fluorescence implying True. Equality of output takes this a step further, and states that all True outputs must be exactly $V_1$.

The third property is *commutability*. An AND gate is an example of a commutable gate; inputs are not separated by moderators and are similar. The inputs could arrive in any order (including simultaneously) during the *input phase* and still operate correctly. The input phase is the section of the pulse order in which the inputs arrive, separate from any moderators or the output pulse. During this phase, the inputs could be rearranged, arrive simultaneously or have a delay between arrival and the gate will still operate. Commutability is desirable as it reduces an implementation's reliance on the order of pulses.

Photochromic logic gates do not require inputs to arrive simultaneously. Inputs arrive serially for non-commutable gates and serial or parallel for commutable gates. There can be an indeterminate length of time between light pulses, allowing for partial execution of gates, temporally separated inputs, or leaving the output of a gate unknown until a later time.

### *3.1.3 Logic Circuits*

A logic circuit is a set of logic gates connected to one another, with one or more defined input nodes and one or more output nodes. We define a circuit as a set of logic gates and nodes as defined in the previous section, but with an extended notation to record how gates are connected.

A logic gate is now defined as [*id, c, (p*), o*] where *id* is a unique identifier used by other gates' input pulses to reference this gate, and *c, (p*)* and *o* are unchanged from the previous definition.

Input pulses in *(p*)* have an altered definition $\{s, \updownarrow_y\}$ where *s* is the 'source' *id* of the input, either an input node *id* or the *id* of a previous gate. The value of the source determines the value of this input. $\updownarrow_y$ is unchanged from the previous definition.

The circuit as a whole hence has a definition [*(i*), (o*), (g*)*] where:

*(i*)* is a set of one or more input nodes. An input node is just an *id*. No *True* or *False* value is assigned to a node as this definition is a general one for all possible input values.

*(o*)* is a set of one or more output nodes. An output node is *s*, where *s* is the *id* of a logic gate or input node in the system.

*(g*)* is the ordered set of zero or more gates in the circuit.

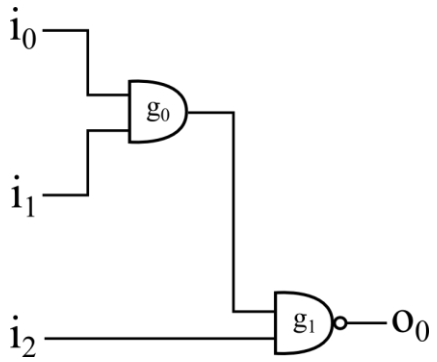



*Figure 3.3: An example circuit* $C_{ex}$.

As an example of this representation, the circuit shown in Figure 3.3 would be represented as:

$C_{ex}$ = [$(i_0, i_1, i_2)$, $(g_1)$, ([$g_0$, 3+, ($\{\underline{i_0, \uparrow_1}\}$, $\{\underline{i_1, \uparrow_1}\}$, $\{\downarrow_1\}$), $\downarrow_1$], [$g_1$, 3+, ($\{\uparrow_2\}$, $\{\underline{g_0, \downarrow_1}\}$, $\{\underline{i_2, \downarrow_1}\}$), $\downarrow_2$])]

Note that gate inputs now carry specific identifiers of their source, rather than the *a, b* of single logic gate definitions. Also note that the output node *id* $o_0$ is not

included in this representation. Instead, the *id* of the gate whose output is the value of $o_0$ is included.

An alternative representation suitable for large circuits with repeated gate types is pre-definition of gates. Rather than defining each gate in *(g*)*, only the identifiers and a gate-type are included, and the gate definitions are given separately. The example circuit in Figure 3.4 would be defined as:

$C_{ex2} = [(i_0, i_1, i_2), (g_1), ([g_0, 2\text{-}NOR, (i_0, i_1)], [g_1, 2\text{-}NOR, (g_0, i_2)])]$

$2\text{-}NOR = [2+, (\{\uparrow_1\}, \underline{\{a, \downarrow_1\}, \{b, \downarrow_1\}}), \downarrow_1]$

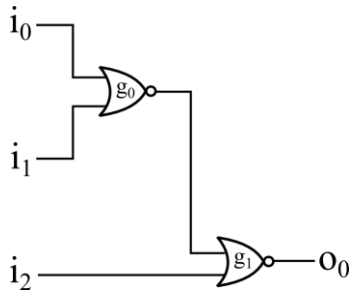



*Figure 3.4: An example circuit* $C_{ex2}$.

## 3.2. Programming the System

Programming a photochromic memory-based system starts with a logic circuit you wish to implement, serialises it into a sequence of gates, and then defines each logic gate as a sequence of light pulses. The implementation of logic circuits supports two modes of operation; drag and drop, and compile. The first is suitable for smaller, simple circuits, and involves manually placing and connecting gate objects in a program. The second is suitable for larger, more complex circuits, and involves designing the circuit in Logisim [Burch (2005)], compiling the circuit into a pulse sequence, and running the pulse sequence through an interpreter program.

Both methods must deal with the inherent limitations of photochromic logic gates and our implementation of them, namely, that we do not implement non-unique gates (we circumvent this with equivalent sub-circuits however), that gates cannot be run in parallel (although in principle this could be achieved) and that the output of eNBIPS gates cannot be used as the direct input to a subsequent gate (the major limitation of our approach).

### *3.2.1 Drag and Drop Circuits*

Drag and drop circuits allow a user to implement simple circuits within the LabVIEW programming environment. Section 4.2.3 describes LabVIEW in more detail, but briefly LabVIEW is a development environment with a visual programming language, designed to interface with external hardware. Virtual Instruments and subVIs are the functions within this language, and they are wired together to exchange data. By programming gates as subVIs (detailed in Section *7.3*), the user can drag and drop gates to construct circuits in any LabVIEW program.

Implementation of circuits using non-unique gates (typically XOR) requires manual alteration to replace the non-unique gate with an equivalent sub-circuit of unique gates (called *uniquifying*). As each gate takes the optical tasks as input and outputs the suspended tasks for subsequent gates, the wiring of gates gives an implicit serialisation. The output of each gate is stored by the LabVIEW computer to be reinterpreted as an input pulse. This addresses the three issues our implementations must consider, but the manual uniquification and manual serialisation are prone to human error on all but the simplest of circuits (the complex wiring can be seen in Figure 3.5). To address this problem, a circuit compiler was produced.

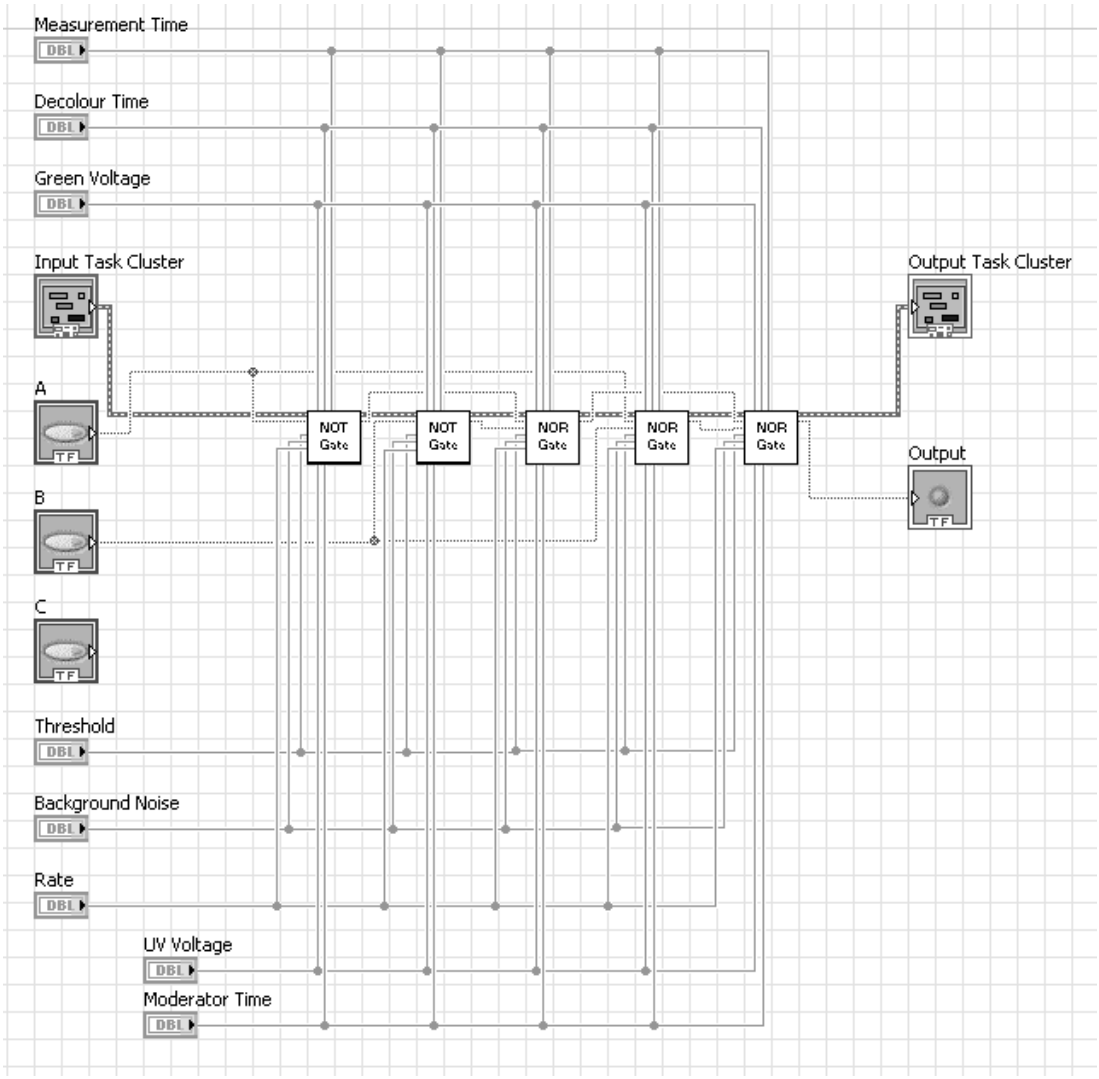



*Figure 3.5: Example drag and drop circuit in LabVIEW. This implements Rule 60 (see* Chapter 8: Photochromic Molecular Elementary Cellular Automata*)*

### *3.2.2 Compiled Circuits*

For larger, more complex circuits for which manual uniquification is unrealistic and manual serialisation would be time consuming, we developed an alternative circuit implementation workflow. Compiled circuits follow a five step process from design to execution.

i. *Circuit Design*: The circuit is first designed in a 3rd party program; Logisim [Burch (2005)]

ii. *Wire Parser*: To convert from Logisim's XML-based .circ file type, a parser was programmed that converts the XML to Java objects.

iii. *Uniquifier:* As we do not implement non-unique gates, the uniquifier was programmed to take the parsed circuit and replace any non-unique gates with equivalent sub-circuits of unique gates.

iv. *Serialiser:* As the planned experimental set-up can only execute gates in serial, the serialiser is a program which calculates the order gates should be executed in. It also outputs a text file with complete instructions for circuit execution.

v. *Execution:* A LabVIEW VI was developed that takes the resulting text file from the serialiser and executes it on our hardware setup.

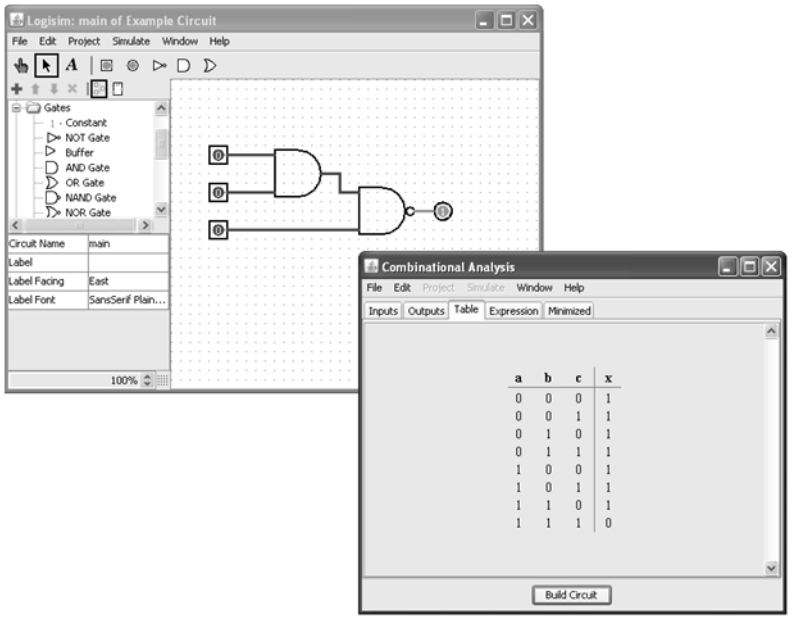
Each step is looked at in more detail.

***Circuit Design***

To design the circuit, we use Logisim [Burch (2005)] as seen in Figure 3.6. Logisim is a tool for designing and simulating logic circuits, offering a simple drag and drop interface and numerous optimisation features. Logisim can generate circuits based on logical expressions, allowing the simple conversion of almost any function into a circuit. It also allows the construction of sub-circuits to simplify the creation of more complex logic gate circuits, and can neaten diagrams to aid interpretation.

Logisim also features automatic circuit minimisation which ensures the circuit is efficient before use. Analysis of the circuit can produce a truth table to ensure the circuit is executing the expected function. These features help save time and materials by ensuring circuit correctness.

*a)*




*b)*

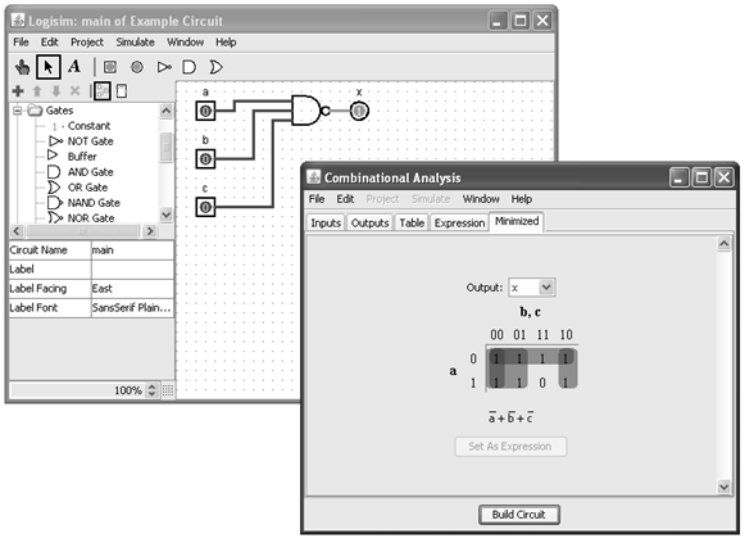



*Figure 3.6: Burch's Logisim. a) Logisim [Burch (2005)] representation of* $C_{ex}$ *(Figure 3.3) and the automatic generation of a truth table. b) Automatic minimisation of the circuit to a single 3-NAND gate.*

***Wire Parser***

The save format of Logisim is the .circ file. This XML-based text file defines the structure of the circuit in a very literal manner using coordinates and attributes for each component in the circuit. The Wire Parser was programmed in Java and uses DOM parsing to turn the XML file into objects, uses the coordinates of the wires in the system to determine groups of connected wires, determine what gate connections are attached to these groups and hence which gate is connected to which and how.

***Uniquifier***

As photochromic logic gates do not have unique implementations of XOR or XNOR logic gates, any circuit must have these gates replaced. The uniquifier replaces XOR or XNOR gates in the circuit with an equivalent sub-circuit of NOR gates (as the simplest universal photochromic logic gate), giving the same functionality but a necessarily longer execution time, shown in Figure 3.7.

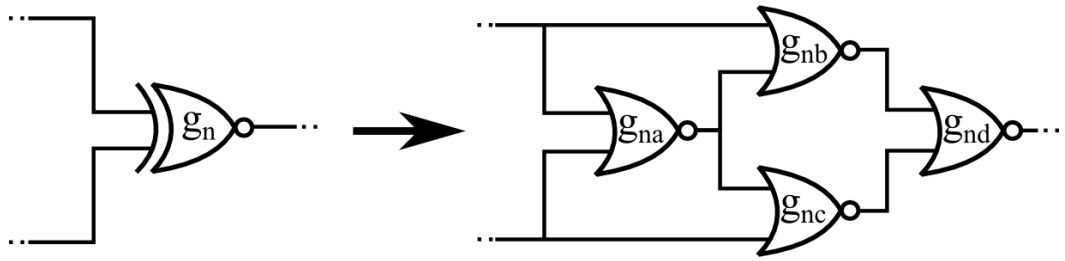



*Figure 3.7: The uniquifier converts non-unique gates (such as an XNOR on the left) into a sub-circuit of NOR gates that implements the same logic, without disrupting the rest of the circuit.*

This is useful as a quick fix for circuits with very few XOR or XNOR circuits, but for more complex circuits with many non-unique gates the removal is better handled as the Logisim stage, as the replacement sub-circuits carry an associated slow down (due to needing to execute more gates). The circuit minimisation functions built in to Logisim offer a more intelligent approach to the removal of redundant gates.

***Serialiser***

The planned hardware implementation can only address a single logic gate at a time. As a result, any circuit with parallel elements needs to be refactored into a serial order. The algorithm for this is as follows:

i. All gates and input nodes have an ‘order’ property $O_x$. This starts unassigned ($O_{null}$). An order is a unique value representing the serial order gates must be executed in.

*ii.* Assign the *n* input nodes an order from $O_0$ to $O_{n-1}$

*iii.* Loop through all gates. If all the inputs to a gate have an order assigned, assign that gate order $O_x$ such that *x* is the lowest order not currently assigned. If one or more inputs have order $O_{null}$, skip.

iv. Repeat loop until all gates have non-$O_{null}$ order assigned.

The serialiser also formats the circuit for the execution step. An example of the format can be seen in Figure 3.8, where the serialiser output for circuit $C_{ex}$ (Figure 3.3) is shown, with the elements explained.

| Pattern Output | Meaning |
|---|---|
| 2 | Highest register state required. |
| 1 | Number of parallel logic gates (max 1 currently) |
| 3 | Number of Inputs |
| 0,1 | Input Nodes: Input Reference, Input Value |
| 1,1 | |
| 2,1 | |
| 1 | Number of Outputs |
| 0,10 | Output: Output Reference, Output Source |
| 3,0,0,UC,1 | Pulses: |
| 4,0,1,UC,1 | Pulse Reference, Logic Gate id, Source |
| 5,0,-1,GD,1 | Reference, Type (UV Colouration, Green |
| 6,0,-1,GO,1 | Decolouration, Green Output), Value Change. |
| 7,0,-1,UC,2 | |
| 8,0,6,GD,1 | |
| 9,0,2,GD,1 | |
| 10,0,-1,GO,2 | |

*Figure 3.8: Serialiser output for circuit* $C_{ex}$ *(Figure 3.3). Note that at this point,* True/False *values have been assigned to the inputs for execution. Pulse references are the ID of the input node or previous gate that determines whether this pulse will be enacted or not. A pulse reference of -1 implies that pulse should always be enacted.*

***Execution***

Lastly, the serialiser output file is specified as the input into the execution LabVIEW program. This program loops, executing each light pulse in the input file in turn, and recording the fluorescence for any output pulses into an array. Any pulse with a pulse reference will perform an array look up to determine the value of the reference. The execution of compiled circuits is abstracted from the logic circuit itself; it is simply a list of light pulses and dependencies.

# Chapter 4: Methods

To explore the possibilities of NitroBIPS *in vitro* or *in vivo*, it is necessary to first study the possibilities it affords us in isolation. This is the purpose of this thesis, to characterise and implement conventional data storage and conventional universal computation with an unconventional biocompatible molecule. A biocompatible molecule that is provably capable of data storage and universal computation would show great promise in synthetic biology applications.

For this purpose, we require stable samples of NitroBIPS and an optical system with which we can illuminate and record fluorescence from these samples. Both need only be capable of establishing proof of concept implementations.

In this section we detail the methods and protocols necessary to implement the experiments in this thesis. There are two major sections; production of eNBIPS samples, and building and characterising the experimental setup to expose the eNBIPS samples to stimulus. The eNBIPS samples and the experimental setup remain constant for all the experiments detailed in subsequent chapters.

## 4.1. eNBIPS Samples

For our experiments we require samples of NitroBIPS to expose to light and to record fluorescence. NitroBIPS requires dissolution in solvent to function, so any samples must be resistant to evaporation in order for the volume and concentration to remain constant. Evaporation over time will alter the concentration of NitroBIPS and hence alter its optical properties. NitroBIPS is solvatochromic and quantum yields

also change depending on the solvent. Good quantum yields at this stage are of a low priority as poor quantum yields simply increase the time to run experiments; the possibilities of NitroBIPS are unaltered.

NitroBIPS is supplied as a powder (273619 Aldrich) and can be dissolved in a variety of solvents, including water, ethanol, methanol and toluene [Görner and Matter (2001)]. Low polarity solvents improve quantum yields but increase the rate of thermal relaxation. An unencapsulated solvent will rapidly evaporate, altering the rates of colouration and decolouration as the concentration of NitroBIPS changes. As discussed in later sections, much of our research relies on NitroBIPS samples behaving consistently over time so a fixed duration pulse of stimulus causes the expected response. As a result, any NitroBIPS samples must be encapsulated.

Samples should be stable to reduce wastage. This requires the samples be kept in the dark to prevent photobleaching. Samples should be uniform so sub-regions can be addressed and expected to operate in the same way as any other sub-region. Samples should also be safe to handle and inexpensive to produce.

Polydimethylsiloxane (PDMS) is a silicon polymer. It is transparent to both visible and UV light [Lee et al. (2003)]. PDMS is non-toxic [Mark (1999)] and is sometimes used as a food additive [McDonald's Corporation (2012)]. Spiropyrans can be encapsulated and remain functional in polymer matrices [Stitzel et al. (2006)] [Atassi et al. (1995)], though with an associated decrease in quantum yields of nitro-substituted spiropyrans (such as NitroBIPS) by inhibiting the isomerization of the molecule. [Dvornikov et al. (1994)] This also causes an associated increase in fluorescent yields due to increased stability of the merocyanine form.

A sample of encapsulated NitroBIPS (eNBIPS) is produced by first dissolving NitroBIPS in methanol (as a PDMS non-solvent [Mark (1999)]) to a 4 mM concentration, mixed in a 5:1 ratio by volume with PDMS rubber (PDMS, Dow Corning Sylgard 184) and the associated curing agent (10:1 ratio PDMS to catalyst). This produced a final concentration of 0.67 mM. This mixture was then poured into a 90 mm diameter petri dish to a 1.15 mm depth (6900 mg weight) and left to cure for 48 hours at room temperature in the dark. The PDMS was left to self-degas to remove all introduced air bubbles. The cured rubber could then be cut into pieces of

the desired size. NitroBIPS is a skin, eye and respiratory irritant, but PDMS is safe to handle. eNBIPS can be handled directly safely.

Clean PDMS will stick to smooth clean surfaces, allowing us to mount eNBIPS vertically. The resulting samples are stable with regards to bleaching; samples stored in the dark for more than one year were found to still be functional (albeit with a reduced dynamic range). Samples left exposed to normal room lighting will bleach however, though at a low rate.

## 4.2. Experimental Setup

A system is required to illuminate and detect fluorescence from the eNBIPS samples. This requires several components working together (illumination sources and detectors) and hence requires a central control mechanism. As discussed in *2.7 Spiropyran Photochromic Molecular Switches and NitroBIPS*; the inputs to the molecules are ultraviolet and visible light, and the output is orange fluorescence. Samples of NitroBIPS are flat pieces of rubber that can be cut to any shape and need not be illuminated in their entirety, but both green and UV illumination must illuminate the same area. Photons emitted by NitroBIPS are emitted in all directions with an equal probability as fluorescence is an isotropic process; the system should detect as large a proportion of these as possible but it need not be every photon. Emitted photons should be captured and turned into an electrical signal for analysis. Some applications also require that the area of illumination be movable.

### *4.2.1 Illumination*

Several possibilities exist for controlled illumination but the most common three used for spiropyran experiments are light-emitting diodes, arc lamps, and lasers.

Light-Emitting Diodes or LEDs are pieces of doped semiconductor, where the impurities introduced during the doping process are different on one end than the other. One end is p-type (i.e. positive, and has additional electron holes) and the other is n-type (i.e. negative, and has additional electrons). The resulting p-n junction releases photons when a current is passed over it as a result of the electrons meeting the electron-holes, moving to a lower energy level, and releasing the remaining energy as a photon. Different dopants produce different wavelengths of photons. Of the three, LEDs are cheapest but are the lowest intensity.

Arc lamps produce an electrical arc in an ionised gas between two tungsten electrodes. This creates a high intensity and stable illumination with a wide spectrum of wavelengths. Arc lamps are moderately expensive, but produce the highest intensity light across a wide band of wavelengths.

A laser causes the emission of photons from a pumped gain medium. A suitable substance is pumped with an energy source such as an electrical current or flash lamp. This causes electrons to move to a higher energy level and emit photons upon decay. The emitted photons are internally reflected with mirrors, causing other atoms in the gain medium to emit photons coherently. A proportion of the resulting light is released through a partially reflective output coupler. The result is a high-intensity beam of monochromatic, coherent and directional light, but lasers are expensive.

All three methods are effective light sources, and have been shown to be suitable candidates for photochromic studies. [Natarajan et al. (1992)] [Levitus et al. (1997)] [Stitzel et al. (2006)] We chose LEDs for our experiments. Laser properties of monochromaticity and coherency are not advantages for our application. Arc lamps provide very high power illumination, but require filtering to provide only the desired wavelength of light, and shuttering to permit accurate control of illumination timing. They also reach high external temperatures. By comparison, LEDs are cheap, easier to operate, operate at a low temperature (NitroBIPS is subject to thermal effects), produce sufficient intensity light [Stitzel et al. (2006)], and are easy to control both in gate and intensity.

Two LEDs are used; one 365 nm UV (specified bandwidth ~350-396 nm) and one 530 nm Green (reported bandwidth ~464-582nm), both from Cairn. The wavelengths were chosen as being within reported absorption spectra for NitroBIPS in toluene, which is expected to be similar to the eNBIPS spectra peak. [Mark (1999)] The LEDs are mounted with a collector lens to collimate their light. They are controlled with a Cairn OptoLED 2-channel Power Supply connected via an eight-pin DIN cable. The controller allows control over the gate and intensity of the LEDs both manually (via dials/switches) or programmatically via digital and analogue inputs. The OptoLED Power Supply also prevents overloading of the LEDs. The LEDs are controlled via two BNC input cables (per LED) to the rear of

the OptoLED box, one each for gate and current. A digital logic-level signal (i.e. 0 or +5V) passed to the gate connector was used to turn the LED on or off. Similarly, an analogue signal (0 to +10V) is passed to the current connector and used to control the current delivered to the LED and hence the LED intensity.

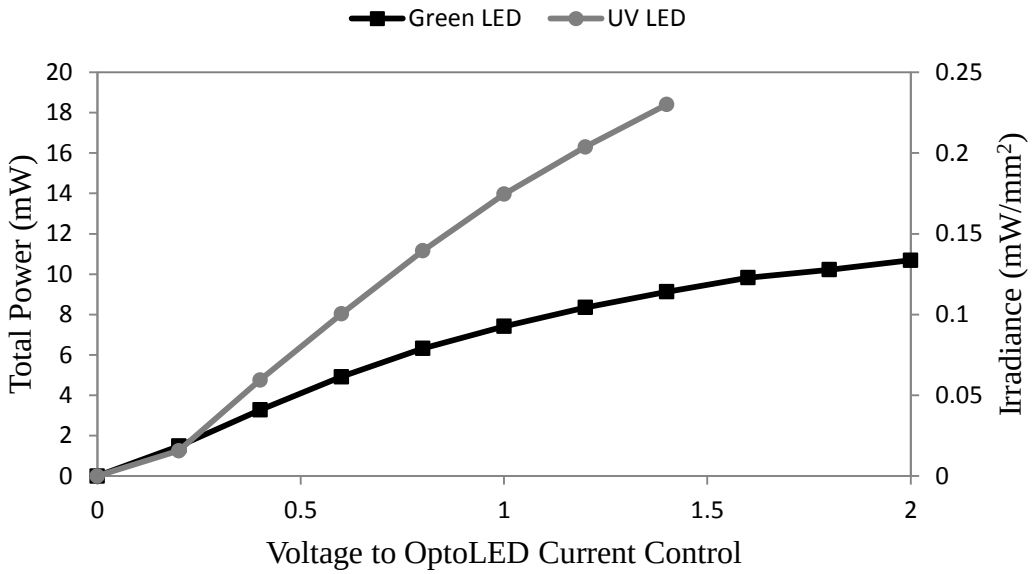



*Figure 4.1: LED irradiance and LED total power.*

The active irradiance and total collected power of the LEDs was measured first, shown in Figure 4.1. The irradiance is the power of light per unit area. It was measured with a 3.25 mm diameter stop (8.3mm area) and a Thorlabs PM100D Light Meter with a S130VC Photodiode. Measurements were taken with increasing voltage until the LED control box reported the LEDs overheating. The green LED had a peak irradiance of 0.14 mW/mm$^2$ at 2 V, compared to the ultraviolet peak irradiance of 0.24 mW/mm$^2$ at 1.4 V. The turn-on time (time from zero to peak intensity) was measured at 37 ms for the green LED, and 35 ms for the ultraviolet LED. The turn on-times were consistent for all voltages. The turn-on time is primarily the delay between a control signal being sent by the control program and to being received by the LED. The turn-off times (time from signal to turn off the LED to the LED reaching zero intensity) were approximately equal to the turn-on times.

The light from the LED is not spatially even due to the filament. The filament produces the light, but not in a completely uniform manner, as seen in Figure 4.2. To rectify this, Köhler illumination is employed. Köhler illumination illuminates the sample with a perfectly defocused image of the LED, creating a uniform disc of light. Two matching Köhler illumination arms were built - one each for the LEDs - that would produce matching discs of uniform illumination. A diagram of the Köhler illuminator can be seen in Figure 4.3. The aperture and field diaphragms respectively allow adjustment of the intensity of illumination manually, and to alter the size of the projected disc. By replacing the field diaphragm with a stop or slit the size and shape of illumination can be adjusted.

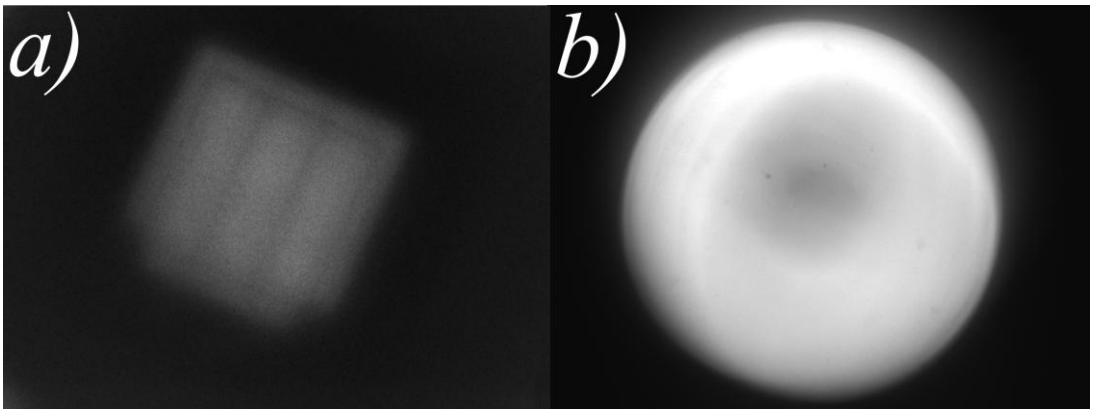



*Figure 4.2: Removing the LED filament from illumination. a) The filament projected by the LED produces non-uniform light. b) Köhler illumination defocuses the image, producing more even illumination.*

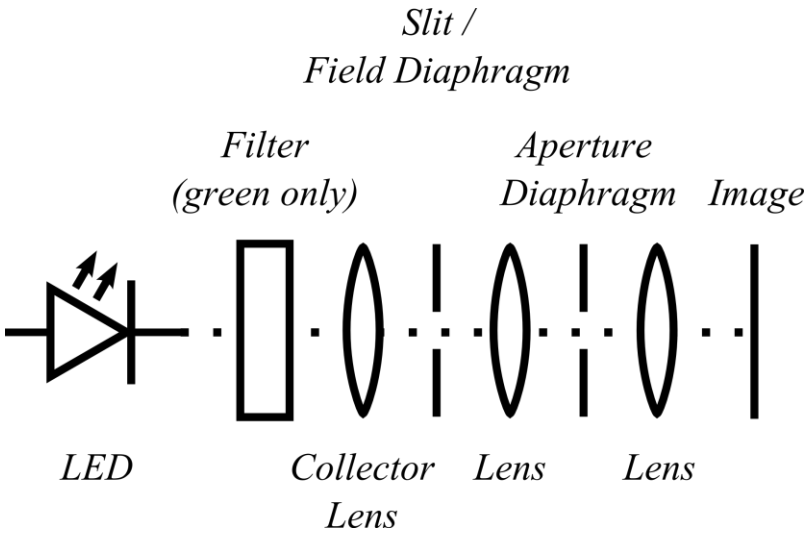



*Figure 4.3: Diagram of the Köhler illuminator. The filter removes wavelengths of light emitted by the green LED that could be conflated for sample fluorescence. The collector lens collimates the light from the LED, the slit or field diaphragm allows control over the size/shape of illumination, and the aperture diaphragm can adjust the intensity of illumination.*

The green LED has a long tail on its emission spectrum, emitting photons that are within the wavelength range to pass through the fluorescence filter. This results in erroneous measurements for fluorescence, as LED photons are recorded by the photodiode. To rectify this, a band-pass filter was placed in the green Köhler illuminator with a band of 507-543 nm. The light efficiency of the Köhler illuminators can be seen in Figure 4.4, representing the difference in irradiance at the LED, and at the end of the arm. The lower UV efficiency is due to glass being partially opaque to UV wavelengths, attenuating the light as it passes through the Köhler illuminator.

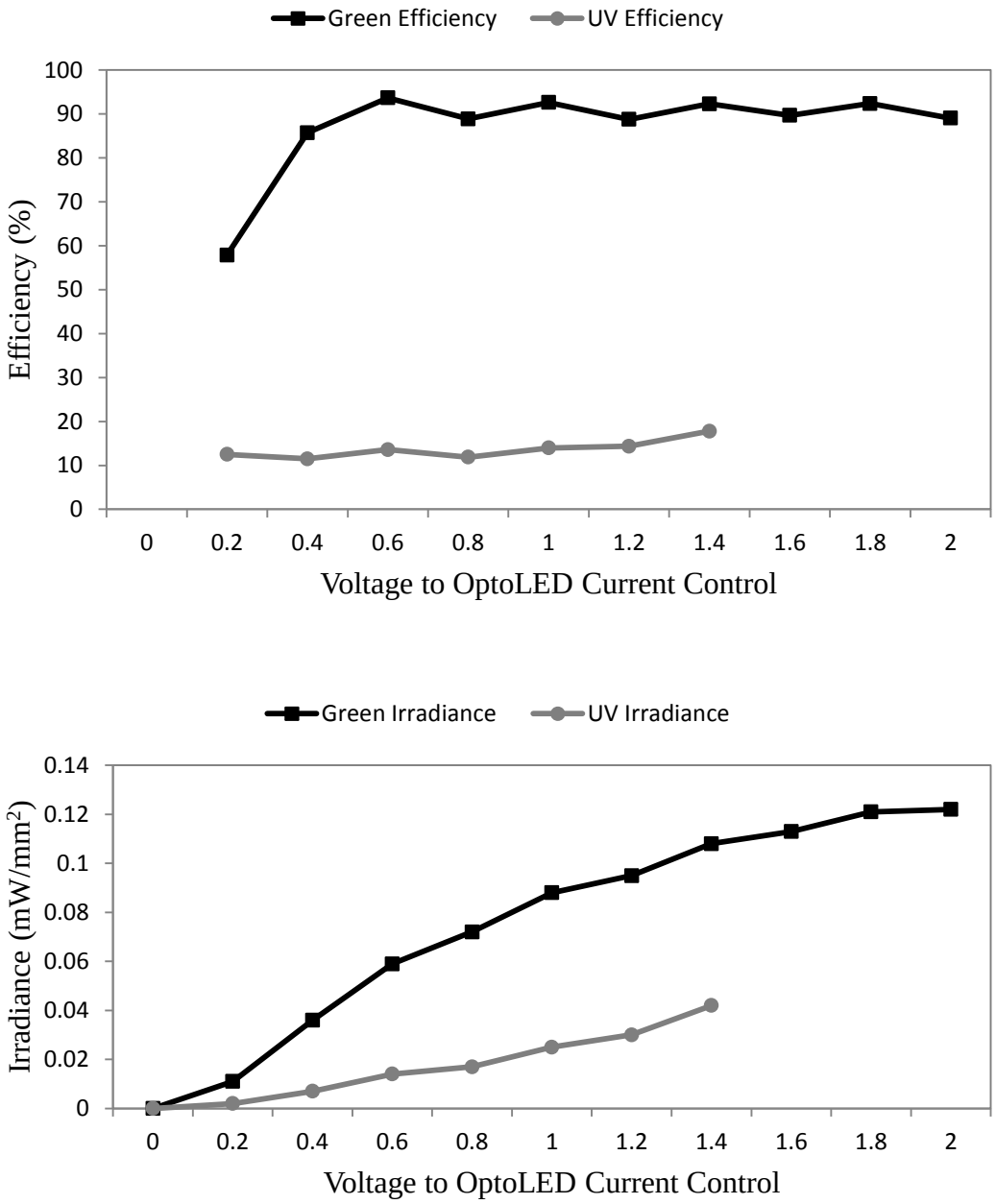



*Figure 4.4: Efficiency of and irradiance after the Köhler illumination arms.*

Each Köhler illuminator produces a separate uniform disc of illumination at its output. These must be combined and relayed onto the sample. This is achieved with a dichroic mirror; a long-pass filter which passes longer wavelengths and reflects shorter wavelengths. A Chroma T495lpxr dichroic filter is used ($d_1$), passing green light and reflecting the UV light. The irradiance and efficiency of the dichroic mirror $d_1$ is shown in Figure 4.5. Lost light is due to scattering, or the absorption of UV photons.

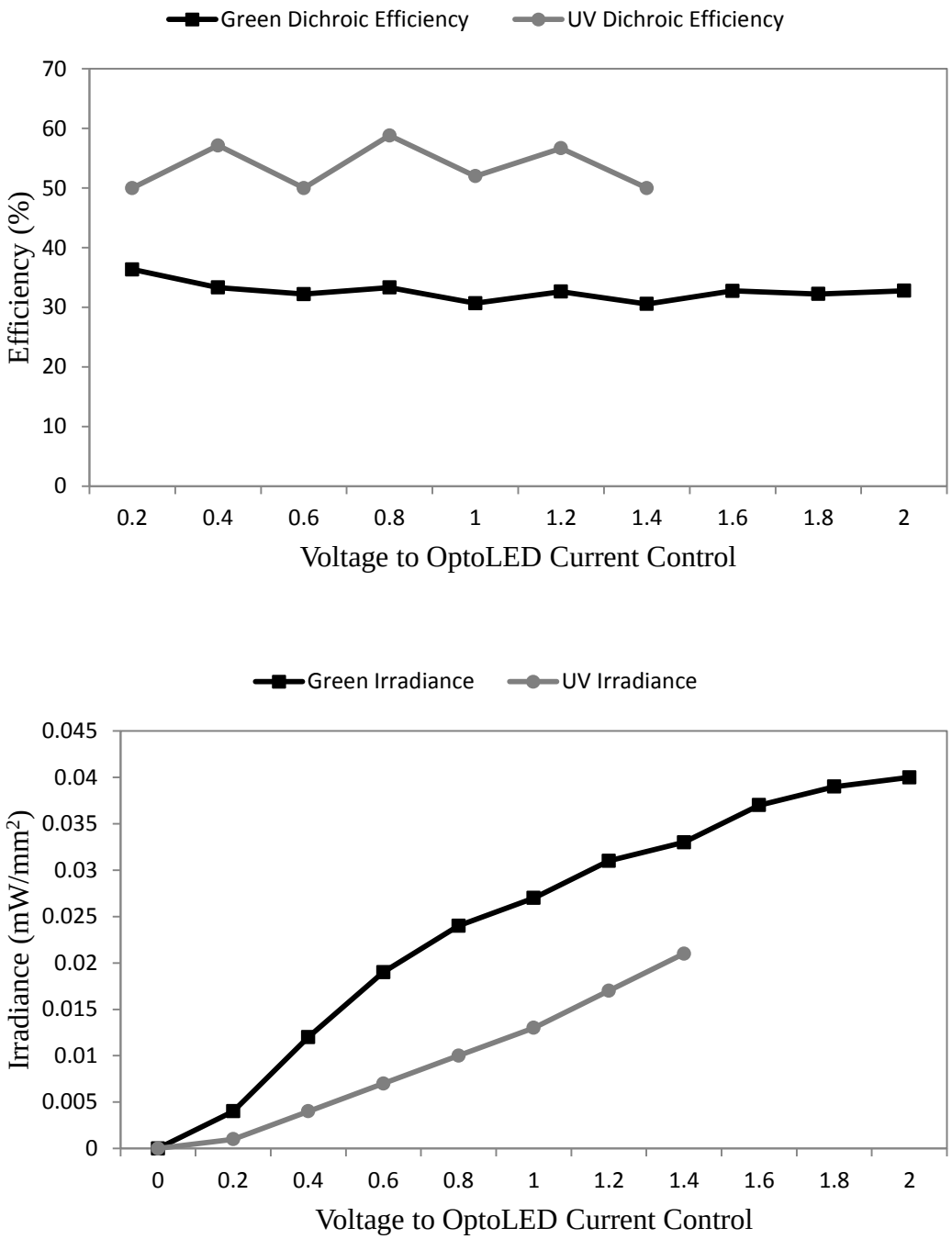



*Figure 4.5: Efficiency of and irradiance after dichroic mirror $d_1$.*

In addition to illuminating the sample, it is also necessary to collect the emitted fluorescence. This light is collected with the illuminating objective lens and directed to the detector via a second dichroic mirror. A Chroma 565DCXR dichroic mirror is used ($d_2$), which passes the 610nm fluorescence of NitroBIPS, while reflecting all the illumination wavelengths. The efficiency of $d_2$ can be seen in Figure

4.6. The light is projected onto the sample with a Nikon M-Plan 20x objective lens 19.9mm from the sample.

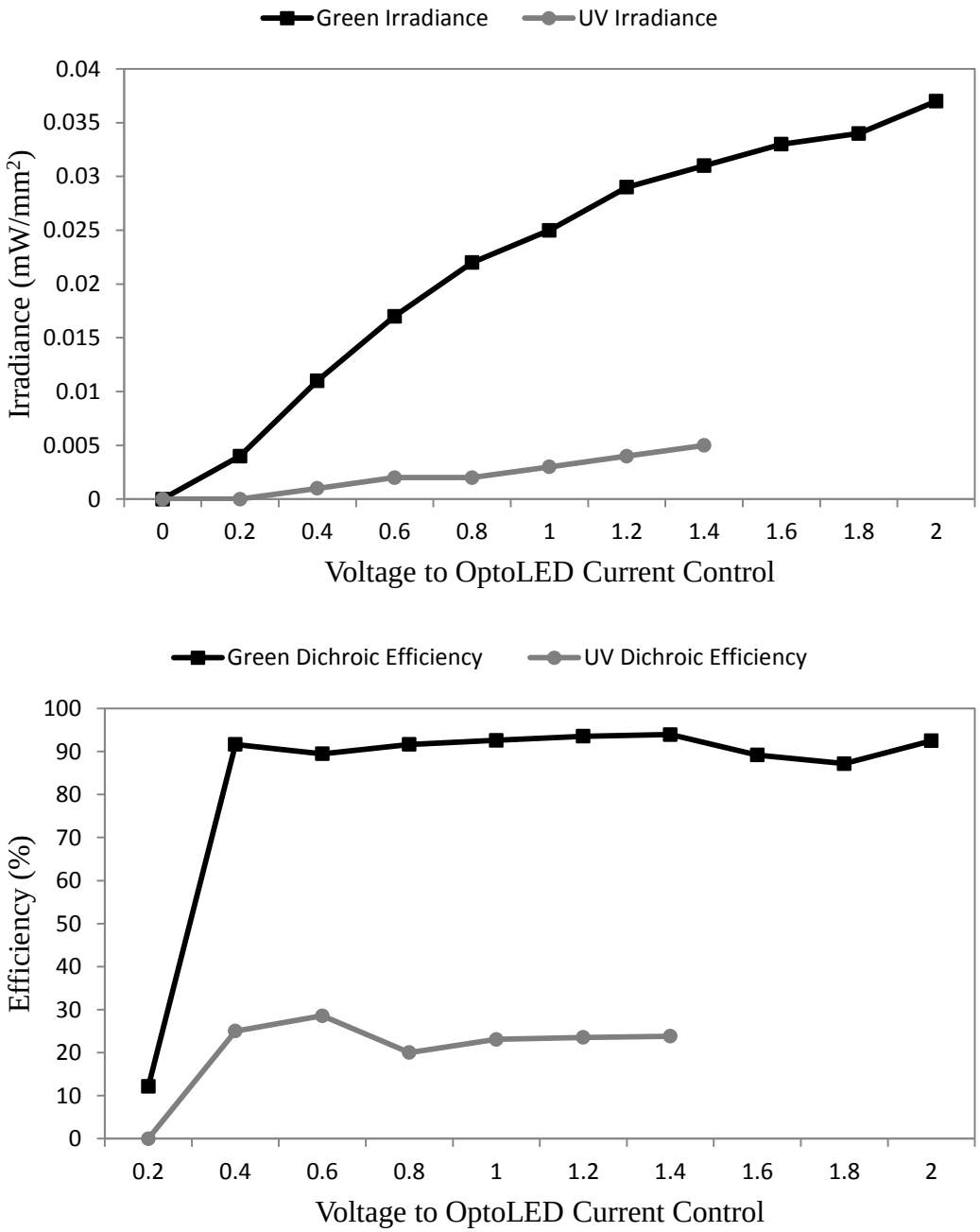



*Figure 4.6: Efficiency of and irradiance after dichroic mirror $d_2$.*

The intensity of light at the sample is shown in Figure 4.7. The peak power for green is 0.974 mW with an irradiance of 1.98 mW/mm$^2$, corresponding to $5.3\times10^{15}$ photons per second per mm$^2$, by the formula:

$$N_{ph} = \frac{\lambda P}{hc}$$ Equation 5

where $N_{ph}$ is the number of photons per second per mm$^2$, λ is the photon wavelength, P is the light power, *h* is Planck's constant, and *c* is the speed of light. This approximation assumes all photons are at the LED's peak wavelength.

Similarly, the peak power for ultraviolet is 0.046 mW, with an irradiance of 0.093 mW/mm$^2$ or $1.7\times10^{14}$ photons per second per mm$^2$. Comparing the efficiency measurements in Figure 4.4, Figure 4.5 and Figure 4.6 shows ultraviolet light is attenuated by optical components, and hence - despite the UV LED having a higher total power than the green LED (as shown in Figure 4.1) - the power at the sample (Figure 4.7) is an order of magnitude lower. Figure 4.8 shows the efficiency of the system. With a green efficiency of ~16% but a UV efficiency of just 0.3%, it is clear improvements could be made. Although relatively low, these intensities are sufficient for the experimental proof of principle in this thesis. The light power could easily be improved significantly in future work.

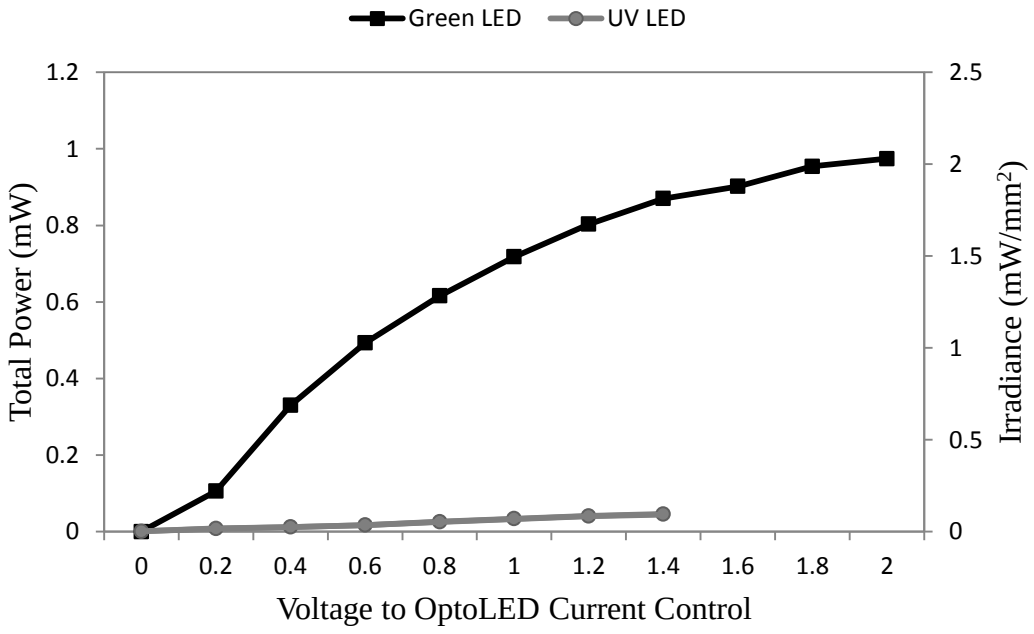



*Figure 4.7: Measured irradiance and total power at the sample. Note that the irradiance (as a measure of light power per unit area) is higher than previous values as the light is focused to a small point.*

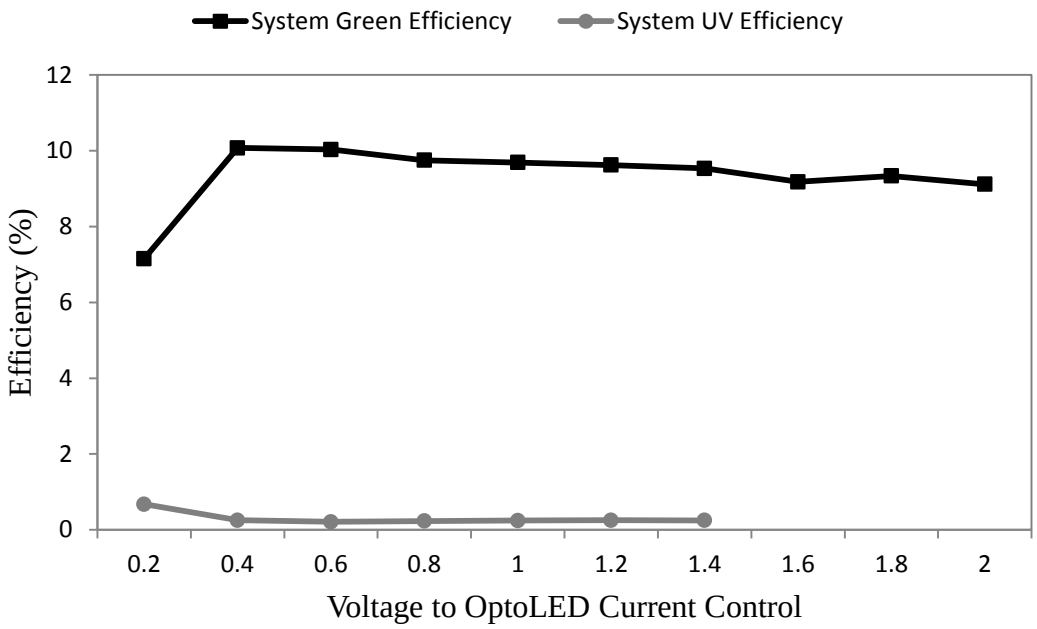



*Figure 4.8: Overall system efficiency. UV light is heavily attenuated as it passes through the system.*

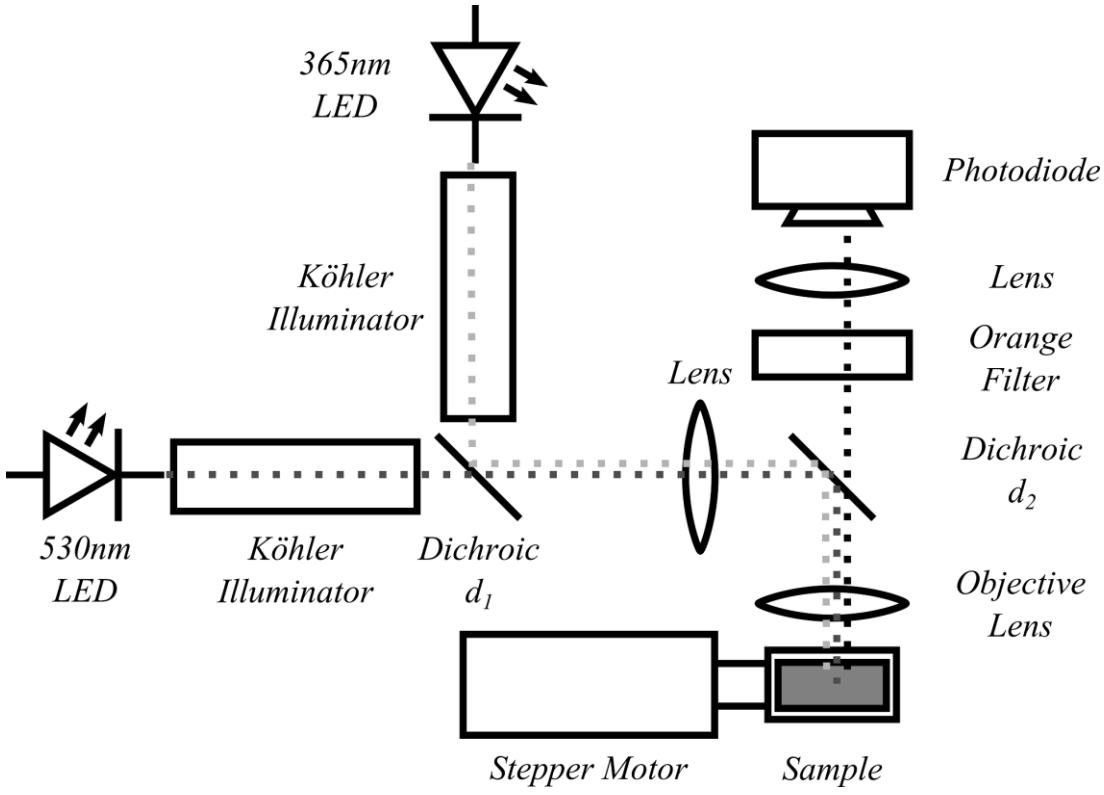



*Figure 4.9: Optical diagram of the complete hardware setup.*

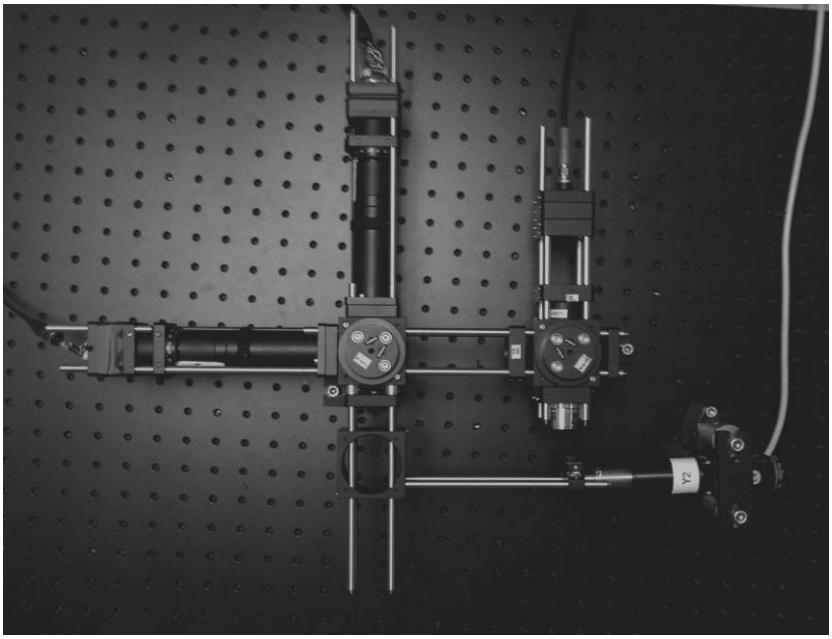


*Figure 4.10: Picture of the completed hardware setup.*

The system requires that the area of illumination be movable relative to the sample to allow addressing of sub-regions on the eNBIPS samples. As molecules are fixed in position by the polymer matrix and hence not subject to diffusion, any sub-region of the homogenous sample can be addressed independently and remain in a state separate from surrounding regions. In order to demonstrate proof of principle it is only initially necessary to control illumination in one axis.

Rather than move the illumination source itself it is simpler to move the sample relative to the illumination. An 8MS00-10-28 model motor from Standa with a travel of 10 mm and a maximum precision of 0.156 µm ($1/8^{th}$ of a step) was chosen. The system requires a high degree of repeatability in positioning to ensure the same sub-region is illuminated. Stepper motors feature non-cumulative error, ensuring accuracy of repeated illuminations. The disadvantages of vibration while moving and low torque characteristic to stepper motors are not issues as the motor will spend the majority of the time stationary and the sample is lightweight. It is mounted as shown in Figure 4.9 and Figure 4.10, moving the sample back and forth over the illumination area.

The uncollected initial intensity of the LEDs is 10-20mW, sufficient to qualify as Risk Group 2 by the European Artificial Optical Radiation Directive

2006/25/EC safety guidelines [Radiation Protection Division, Health Protection Agency (2007)]. Risk Group 2 states that the source is mostly safe due to natural human aversion to bright lights, and that the blinking reflex is quick enough to avoid retinal damage. As a large proportion of the light emitted by the UV LED is invisible to human eyes, exposure may not be obvious and may not trigger natural human aversion increasing the risk of photoretinal damage. The UV LED is additionally a UVA source of an intensity that can also cause skin erythema (reddening and swelling of skin) with prolonged exposure (hours).

The optics are shrouded with a black cloth to ensure light does not escape. This also serves to reduce environmental light that might foul measurements. Due to the increased hazard of the UV LED, the UV LED was turned off where possible, and UV protective goggles were used whenever exposure to UV light was a possibility.

### *4.2.2 Fluorescence Detection*

The detection of light is the primary output from the experiments, so represents a crucial part of our experiments. A photodiode is almost identical to an LED, consisting of a doped semiconductor with a p-type and n-type end. Whereas an LED converts electrons to photons, a photodiode converts photons to electrons, producing a measurable photocurrent. Rather than use a calibrated and expensive light meter (such as the ThorLabs PM100D used for irradiance measurements), a separate photodiode was used that was permanently mounted in our system.

The photodiode is a FDS100 model from ThorLabs. It is sensitive in the 610 nm range we require [Wohl and Kuciauskas (2005)] with a 0.4 A/W responsivity. The output is amplified by a ThorLabs PDA 200 C transconductance amplifier with an amplification range from $1\times10^{3}$ to $1\times10^{8}$ V/A. The Photodiode Amplifier has an analogue output which we can use to record the measured fluorescence.

A Benchmaster 8.41 8-pole pulse hardware filter from Kemo was also used. It allows for both the filtration of noise, as well as further signal amplification up to 500x. The ability to filter noise and the additional amplification facility coupled with the amplification on the photodiode amplifier gives us a sufficient signal to noise ratio to work with our samples.

Fluorescence emission is isotropic so NitroBIPS molecules emit with an even probability in all directions. As a result, only a proportion of photons emitted will be collected by the lens and fewer will reach the photodiode. The proportion of collected photons is given by the formula $\Omega / 4\pi = (1 - \cos\theta)$ where Ω is the solid angle and θ is half the angle subtended by the lens, which is itself given by $\theta = \sin^{-1}(NA/n)$ where *NA* is the numerical aperture of the objective lens and *n* is the index of refraction. With an objective lens with a numerical aperture of 0.35 operating in air (and hence an index of refraction of 1), this subtends a half angle of 20.5° and hence collects 6.3% of emitted photons.

### *4.2.3 System Control*

Interface with the OptoLED control box and the hardware filter is with 0 - +10 V analogue signals connected with BNC cables. The stepper motor control unit connects via a USB cable. The need for programmable system control mandates the use of a PC as the basis of system control. The variety of tasks the system needs to perform requires a programming language to implement system control. The solution needs to send and recieve signals via BNC cables, so additional hardware to allow BNC connections and a way to programmatically control the signals are also required.

National Instrument's LabVIEW (Laboratory Virtual Instrumentation Engineering Workbench) is a development environment featuring a visual programming language (called 'G', but commonly referred to as 'LabVIEW code') and dedicated hardware for interfacing with external instrumentation. As the industry leader in system design platforms, many instruments include code for interfacing with LabVIEW. As National Instruments provides breakout boxes with BNC connectors, and as the stepper motor controller includes LabVIEW control code, a LabVIEW system is our chosen solution.

An NI PCI-6221 card and a BNC-2090A breakout box has enough analogue and digital output signals to control the OptoLED control box, and an analogue input to record the photodiode signal. The PCI-6221 card has an analogue input update rate of 250 kilosamples/second (kS/s), an analogue output update rate of 833 kS/s, and a digital I/O clock rate of 1 MHz.

G is a visual programming language, designed to be used by those without traditional programming experience, though this belies the depth and potential complexity of G code. G code is used to produce Virtual Instruments (VIs) consisting of a wiring panel which determines the VI's function, and a front panel where controls and outputs are placed. In combination with an extensive library of pre-programmed sub-VIs, G is more than flexible and powerful enough to implement any foreseeable data acquisition and control program required.

NI-DAQmx is National Instrument's latest data acquisition driver suite, compatible with LabVIEW but also C++ and other programming languages. DAQmx simplifies the acquisition and generation of analogue and digital signals. Control of the components in our system are with four DAQmx tasks; collections of channel information, timing and control parameters which are referenced when any optical element is controlled. One task is used for each of the LED gate, the LED voltage, the photodiode measurement and a controlling clock which synchronises the tasks. Each LabVIEW program must initialise these tasks and send the task references to any subVI which requires control over these elements. Tasks are started whenever any optical task begins and suspended at the conclusion, until they are woken by a subsequent subVI.

Control of the stepper motor is via supplied LabVIEW device drivers. The included LabVIEW code was designed to be used manually rather than programmatically, so it was modified to be addressable by other LabVIEW code (essentially simulating button presses) and also to include an auto-setup routine. This automates the setup process, including connection to the hardware, resetting to the limit bound, setting maximum safe operating temperatures and voltages, and setting the speed and step size.

### *4.2.4 LabVIEW subVIs.*

Several sub-VIs were written that implement primitive operations that are likely to be repeated, such as illumination and moving the sample with the motor. Each implements a key operation that will be reused during our experiments. Combination and extension of these subVIs implement all the experiments in this thesis. Four subVIs are described here, as shown in Table 2.

| Name | Purpose |
|---|---|
| `Init Optics` | Initialises system connections to optical equipment (LEDs and photodiode) and synchronises operations to an internal hardware counter. |
| `Colourise` | Activates the UV LED for a specified duration and with a specified intensity. |
| `Decolourise` | Activates the green LED for a specified duration and with a specified intensity. Also records the level of recorded fluorescence from the photodiode. |
| `Move` | Moves the area of illumination with the stepper motor to a specified absolute destination on one axis. |

*Table 2: The four key subVIs and their purpose in the system.*

Each described subVI is shown with its high-level representation and the input and output variables. The values for the inputs are not necessarily the values used during experiments, but are of the correct data type.

***Initialise Optics Tasks***

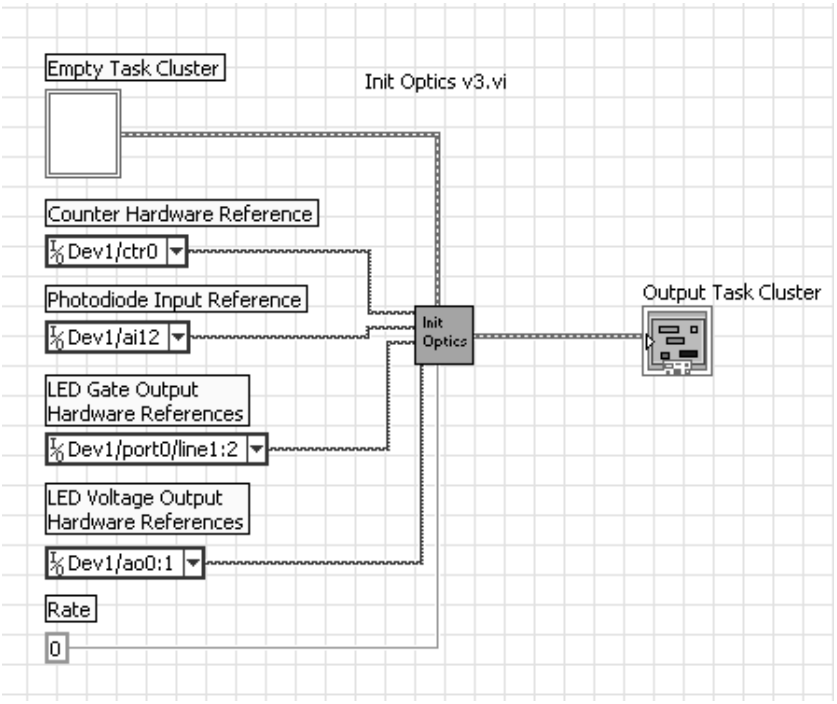



*Figure 4.11:* `Init Optics v3.vi`*, its inputs and its outputs, as shown in LabVIEW. This third-iteration subVI initialises the four DAQmx tasks for future subVIs to access.*

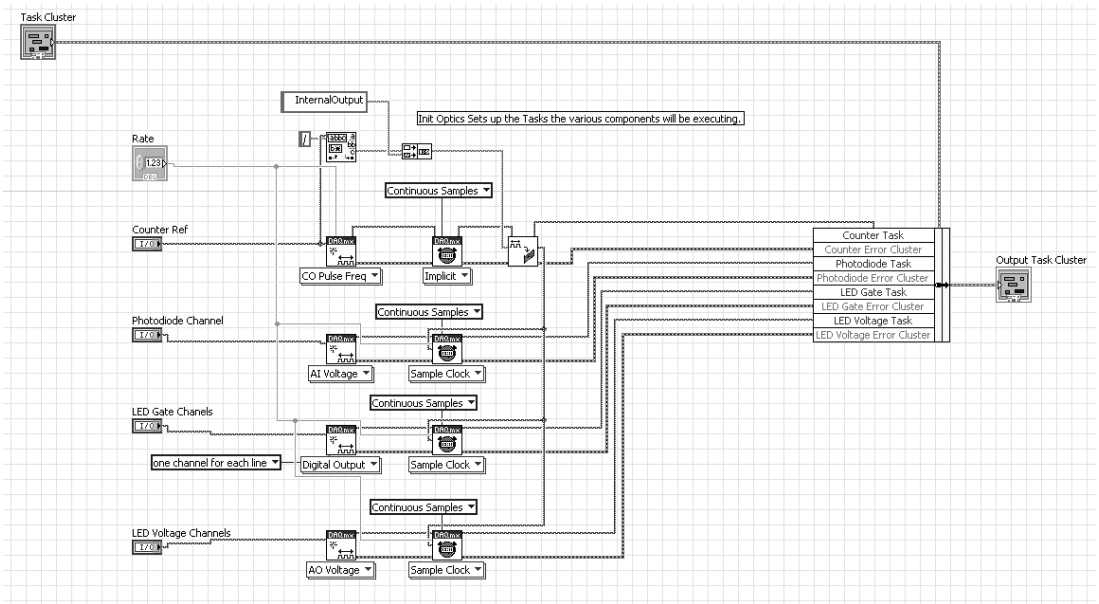



*Figure 4.12: Part of* `Init Optics` *code. The four DAQmx tasks have their type set (analogue input, digital output etc) and synchronised to the counter.*

Every LabVIEW program that interfaces with the hardware setup needs to initialise four DAQmx tasks. These four tasks include hardware references and error clusters. The `Init Optics` subVI, shown in Figure 4.11 and Figure 4.12, handles this. It takes as input:

- `Task Cluster Handle`: This will store the DAQmx task data for each of the four tasks, as well as an error cluster that will save error information in case of hardware failure. A DAQmx task includes virtual channels, timing and triggering information, and other properties regarding the acquisition or generation of data. Other subVIs can then use these tasks to interface with the hardware, either sending data or receiving data.
- `Counter Hardware Reference`: The LabVIEW card includes hardware counters which other tasks will refer to, to synchronize input and output analogue and digital signals. This specifies which counter to use.
- `Photodiode Input Reference`: This specifies which of the analogue input connections on the LabVIEW breakout box will receive the amplified and filtered signal from the photodiode.
- `LED Gate Output Hardware References`: This specifies which of the digital output connections on the LabVIEW breakout box will trigger the gate signal for the LEDs.
- `LED Voltage Output Hardware References`: This specifies the analogue output connections on the LabVIEW breakout box that will output the intensity signal for the LEDs.
- `Rate`: The rate is the sampling rate for the hardware (in samples per second). This is how many time a second a measurement is recorded, or the output value changes.

This subVI creates a sample clock based on the hardware counter and the rate. It then specifies virtual channels for the remaining tasks, specifying the type (analogue output for LED voltage, analogue input for the photodiode, and digital output for the LED gate), assigning the hardware references and associating them

with the counter for synchronisation. Tasks remain in a suspended state until called by a subsequent subVI.

Each task also has an error cluster associated with it, storing the task status (good or error), an error code, and the source of that error. The error clusters allow debugging of hardware failures and for the orderly exit from code upon an error condition. LabVIEW is designed to operate with hardware in the real world, so handling errors gracefully is an essential feature.

The output of the subVI is the completed task cluster, consisting of the task references and error clusters for each task. This cluster is passed to all subVIs which require hardware access.

***Colourise***

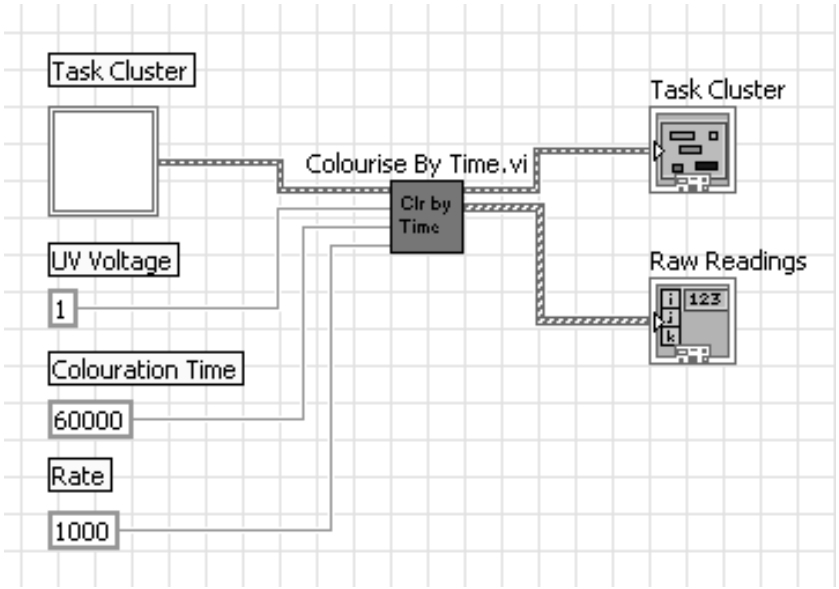



*Figure 4.13:* `Colourise By Time.vi`*, its inputs and its outputs, as shown in LabVIEW.*

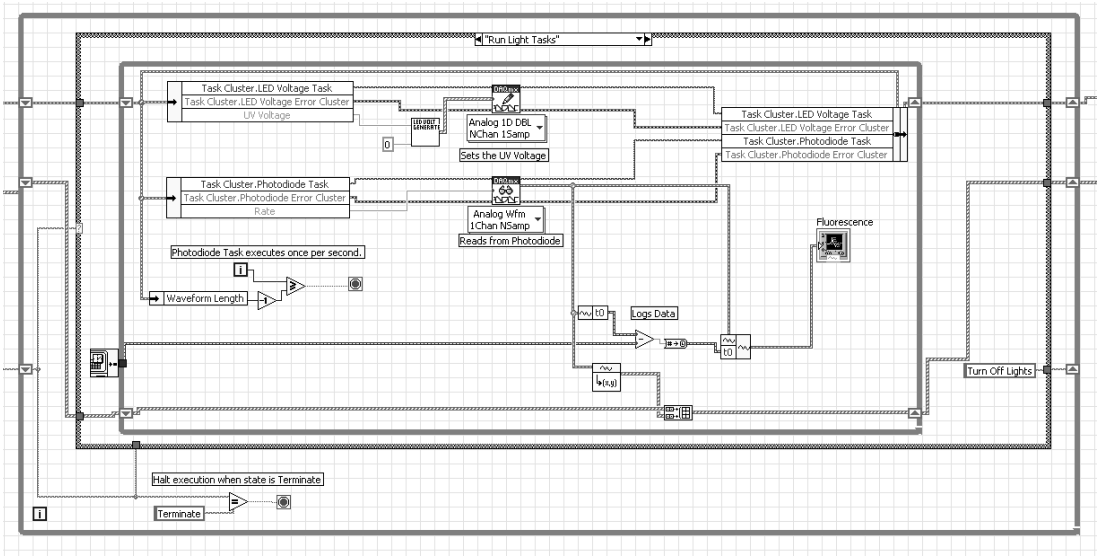



*Figure 4.14: Part of* `Colourise by Time` *code. The code loops, maintaining a high digital signal to the UV LED for the specified time, and reads in data from the photodiode.*

`Colourise By Time.vi` is the basic subVI by which our system interacts with the UV LED. It is shown in Figure 4.13 and Figure 4.14. It specifies the intensity and duration of an ultraviolet pulse. The inputs to the subVI are:

- `Task Cluster`: The hardware references and timing information specified during the `Init Optics` subVI.
- `UV Voltage` (in volts): The voltage passed to the LED power supply, and determines the intensity of illumination.
- `Colouration Time` (in milliseconds): The duration of the pulse. This does not account for LED turn-on and turn-off delay introduced by software and hardware.
- `Rate`: The number of samples buffered in the hardware buffer at a time. As the colourise subVI does not rely on readings from the photodiode, this is typically left at 1000.

The `Colourise` subVI creates a digital waveform with a high-level equal to the colouration time, and creates a flat analogue signal equal to the UV voltage. The subVI then starts all the hardware tasks which results in the two signals being sent to their respective channels to the LED power supply. The subVI then loops for the duration of the pulse, periodically reading photodiode data from the hardware buffer

and logging it. If the clock rate and the rate input match, this corresponds to emptying the buffer once every second. At the end of the colourise pulse the LED voltage and gate are set to zero to turn the LED off, and all tasks are suspended. The subVI then outputs the recorded readings from the photodiode (with timestamps), and the suspended tasks for future use.

***Decolourise***

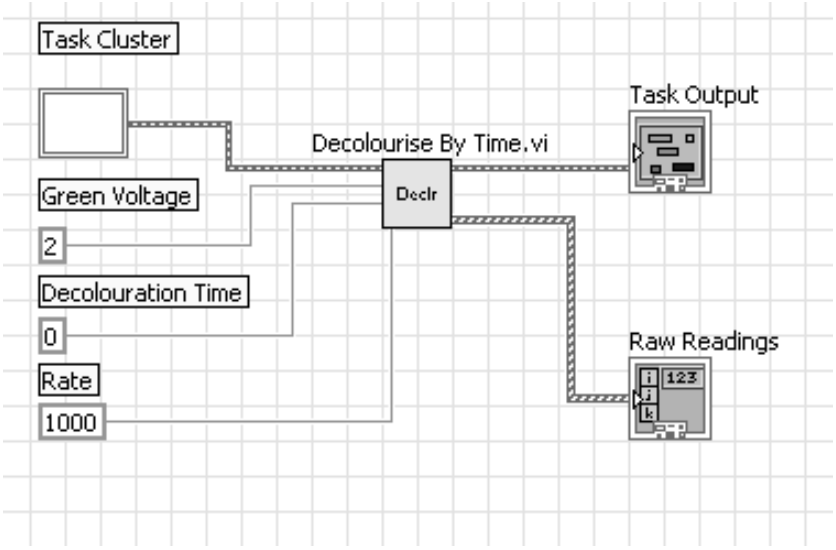



*Figure 4.15: `Decolourise By Time.vi`, its inputs and outputs, as shown in LabVIEW.*

`Decolourise By Time` is the counterpart to colourise, shown in Figure 4.15. It allows our system to enact decolouration pulses of a specified duration. It operates in exactly the same way as described for the Colourise subVI, but dispatches signals to the green LED rather than the UV LED.

The specified `Rate` is more important for decolouration subVIs. `Init Optics` determines how many samples should be generated and buffered each second, and the decolourise subVI `Rate` is how many samples should be buffered before the buffer is read. As a result, lower rates cause the buffer to be read more frequently, higher rates less frequently. Though `Decolourise By Time` does not alter its behaviour based on recorded fluorescence, later extensions to the subVI will; requiring a low rate to read fluorescence frequently.

***Move***

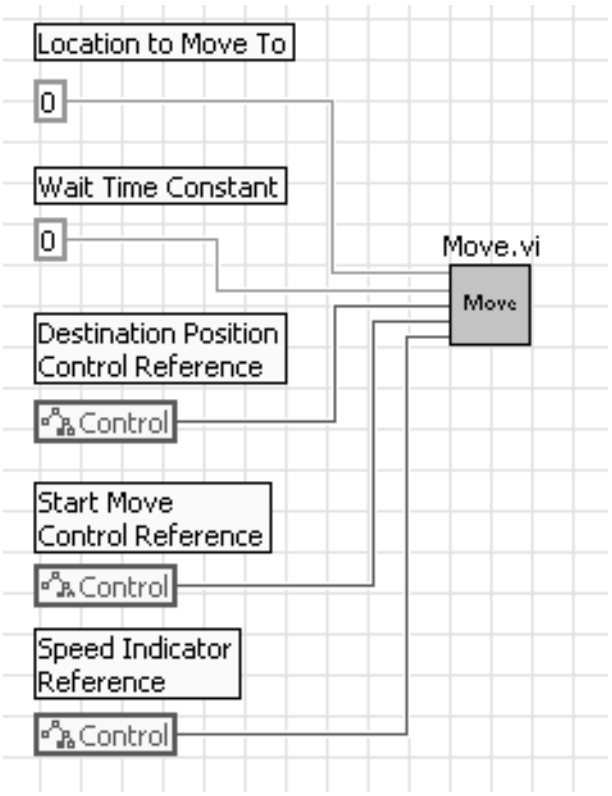



*Figure 4.16: `Move.vi`, an interface between our programs and the LabVIEW drivers supplied with the stepper motor.*

The stepper motor was chosen in part because the motor control was supplied with LabVIEW drivers. However, the drivers were geared towards to manual user interaction. Rather than write new drivers, `Move.vi` simulates a user inputting a destination and starting the movement via LabVIEW control references, which allow user interface elements to be controlled programatically. It has five inputs, as shown in Figure 4.16:

- `Location to Move To`: The destination position for the motor. This is an absolute destination; the motor has a 10 mm travel, so a valid destination is anywhere from 0 mm (fully retracted) to 10 mm (fully extended).
- `Wait Time Constant`: Speed Indicator polling time.

- `Destination Position Control Reference`: A LabVIEW reference to the destination position field in the motor driver.
- `Start Move Control Reference`: A LabVIEW reference to the start move button in the motor driver.
- `Speed Indicator Reference`: A LabVIEW reference to the motor driver speed indicator.

The subVI simulates a user entering the desired destination position and pressing the start move button by invoking them via control references. As the driver does not report when the movement has ended, we poll the speed indicator according to the wait time, and suspend the rest of program execution until the motor has finished moving.

## 4.3. Summary

This chapter has detailed the creation of the experimental setup used throughout the thesis. By encapsulating NitroBIPS in a polymer matrix, we create sheets of homogenous encapsulated NitroBIPS (eNBIPS) which remain switchable and are not susceptible to evaporation or diffusion, allowing for the addressing of sub-regions. An optical system is created and characterised to expose the eNBIPS samples to controlled pulses of ultraviolet and visible light stimulus, and to record the levels of fluorescence emitted by the sample.

The following chapter, *Photochromic Molecular Registers*, details the first steps to implementing conventional logic in NitroBIPS by storing data as the relative proportion of SP and MC-form molecules within a region of eNBIPS. It also looks more closely at effects of optical stimulus on eNBIPS samples and the mathematical principles which govern the rates of colouration, decolouration and fluorescence.

# Chapter 5: Photochromic Molecular Registers

To demonstrate the possibilities of eNBIPS and photochromic molecules as a computational medium, we aim to implement a conventional logic paradigm which has three fundamental components; the register, the logic gate, and logic circuits. In this section the principles of an eNBIPS register are discussed, followed by the experimental procedure required to implement the register.

Different to conventional registers, reading a photochromic register is destructive and will alter the value. Additionally, reading a photochromic register is subject to noise and the value of a register will change over time as the sample thermally relaxes. This chapter details several methods to overcome the destructive nature of photochromic register reads, and demonstrates how to make registers resistant to noise and thermal effects.

## 5.1. Introduction

The ability to store data is an important component in any computing paradigm, even those implemented with unconventional materials. There are a vast number of possible methods in which data can be stored, from the conventional magnetic platters of a hard drive [Goddard (1970)], to the biological code of DNA [Church et al. (2012)], or write-once-read-many properties of thin films of polymer [Lin and Ma (2008)]. All these are intended for the storage of large amounts of infrequently accessed data.

A register is a simple data storage unit for storing smaller amounts of data for shorter periods in a rewritable fashion. The capability for spiropyrans to switch between two forms suggests that data could be stored as the relative concentration of SP and MC form molecules. A register could be switched between states with colouration and decolouration stimuli, and read using excitation stimuli, summarised in Figure 5.1.

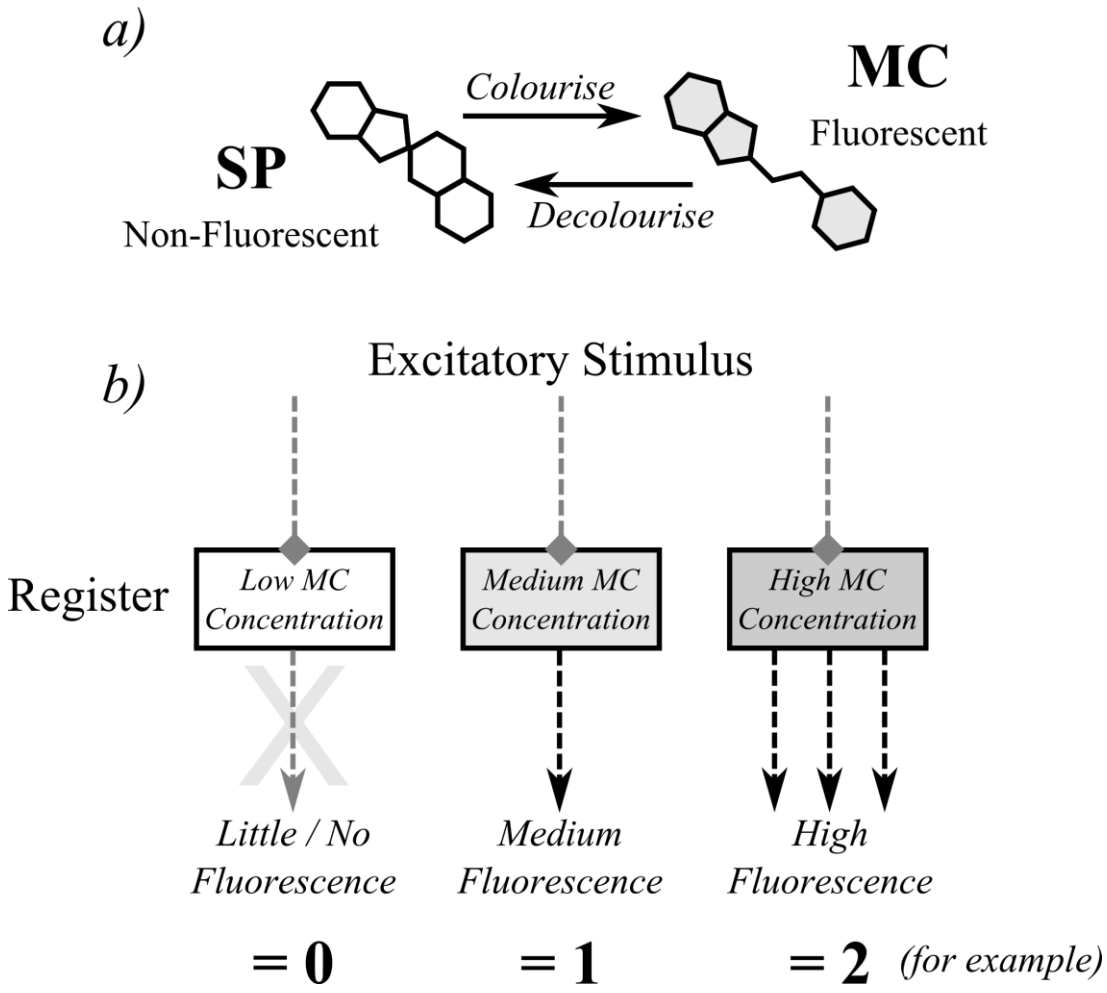



*Figure 5.1: Overview of photochromic register operation. a) An eNBIPS sample can be switched between a fluorescent and non-fluorescent form via the application of colouration and decolouration stimulus, allowing us to write to a register with light. b) The concentration of the MC form in a sample determines the amount of fluorescence when exposed to excitatory stimulus. This corresponds to the value of the register.*

Though *Chapter 4: Methods* describes the hardware to colourise and decolourise eNBIPS samples, registers are the first application in which the colouration and decolouration must be precisely controlled. Data is stored as the

relative proportion of the fluorescent and non-fluorescent forms of NitroBIPS, such that excitation stimulus causes the emission of more fluorescence the higher the proportion of fluorescent-form molecules. As a result, the level of fluorescence can be corresponded to the value the register is storing.

## 5.2. Optical Theory

This section discusses the theory behind switching molecules and we apply elements of this later in this chapter.

The optical density (OD) of a sample is the number of orders of magnitude light is attenuated by when passing through the sample due to photons being absorbed. It is given by:

$$OD = \log_{10}\left(I_\lambda^0 \Big/ I_\lambda^T\right) \qquad \text{Equation 6}$$

where $I_\lambda^0$ is the intensity of light of wavelength λ entering the sample, and $I_\lambda^T$ is the intensity of transmitted light. As NitroBIPS molecules absorb light, it follows that increasing the concentration of NitroBIPS in eNBIPS will increase the OD. In addition, increasing the thickness of the sample will also increase the OD as light has more opportunities for absorption. This phenomenon is described by Beer-Lambert's law:

$$OD = \varepsilon_\lambda^x l c_x \qquad \text{Equation 7}$$

where $\varepsilon_\lambda^x$ is the molar absorptivity of SP or MC-form molecules at wavelength λ, $l$ is the path length (i.e. the thickness of the sample) and $c_x$ is the concentration of SP or MC-form molecules. The molar absorptivity is a measure of how strongly molecules absorb light. NitroBIPS absorbs with a different strength depending on the form of the molecule and wavelength of light. Reported molar absorption coefficients for NitroBIPS are $\varepsilon_{SP}^{300nm}$ = 17,000 $M^{-1}$ $cm^{-1}$ and $\varepsilon_{MC}^{530nm}$ = 28,000 $M^{-1}$ $cm^{-1}$ [Song et al. (2002)]. Measured OD will be slightly lower than calculated with these figures for ε, as the LEDs emit a band of wavelengths which NitroBIPS absorbs less strongly.

An absorbed photon does not guarantee a change of molecular form. The constant probability of an absorbed photon causing an event is defined as the

quantum yield, and is dependent on the solvent, with values for $\Phi_c$ varying from 0.17 in ethanol, to 0.83 in toluene [Görner and Matter (2001)].

A sample of eNBIPS has a concentration $c_{NB}$ (0.67mM for our samples), but this encompasses several sub-concentrations of NitroBIPS molecular forms. $c_{NB} = c_{SP} + c_{MC} + c_{MC*} + c_B$. The fluorescent lifetime of the excited MC*-form molecules $\tau_{MC*}$, and the rate of bleaching $dc_B/dt$ are negligible compared to the duration of illumination so we assume $c_{NB} = c_{SP} + c_{MC.}$ Rates of colouration and decolouration are non-linearly dependant on the concentrations $c_{MC}$ and $c_{SP}$. As colouration is the conversion from SP to MC form molecules, during colouration the concentration $c_{SP}$ drops. As the rate of photon absorption is dependent on the OD and hence on $c_{SP,}$ and as only photons absorbed by SP-form molecules cause colourations, the rate of colouration decays exponentially. Similar for decolouration, as $c_{MC}$ decreases, the rate of decolouration also decreases exponentially. Figure 5.2 shows how colouration and decolouration rates change over time.

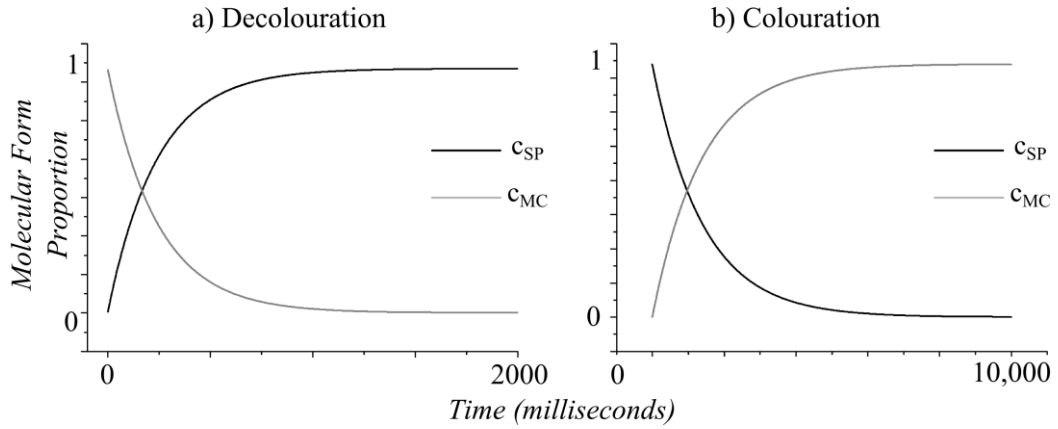



*Figure 5.2: Rates of colouration and decolouration, and their effect on the concentrations of SP and MC-form molecules. a) Graph of* $c_{MC}$ *and* $c_{SP}$ *over time during decolouration, as given by Equation 10 and Equation 11. A typical value for* $k_1$ *was used: 4* $s^{-1}$*. b) Graph of* $c_{SP}$ *and* $c_{MC}$ *over time during colouration, as given by Equation 12 and Equation 13. A typical value for* $k_2$ *was used: 0.071* $s^{-1}$

To quantify these rates, first take a sample of eNBIPS with molar concentration $c_{NB} = N_{NB} / N_A V$ where $N_{NB}$ is the number of NitroBIPS molecules, $N_A$ is Avogadro's number and $V$ is the volume given by $V = Al$ where $A$ is the area of the sample illuminated and $l$ is the light path length. During decolouration it is

assumed the sample is illuminated with a constant, monochromatic and uniform light source of wavelength $\lambda_d$ and irradiance $I_d^0$, over the area $A$ for a short time interval $\Delta t$. This number of photons is given by $N_{ph}^0 == I_d^0 \lambda_d A \Delta t / hc$ where $N_{ph}^0$ is the number of photons incident to the sample, $h$ is Plank's constant and $c$ is the speed of light.

Of those incident photons, the MC-form molecules will absorb a fraction in accordance with the Beer-Lambert law, $N_{ph}^{ab} \Big/ N_{ph}^0 = 1 - T = 1 - 10^{-\varepsilon_{MC}^{\lambda_d} c_{MC} l} = 1 - e^{-\ln(10)\varepsilon_{MC}^{\lambda_d} c_{MC} l}$ and the number of molecular-form changes from MC to SP is given by $\Delta N_{MC} = -\Phi_d N_{ph}^{ab}$ where $N_{ph}^{ab}$ is the number of photons absorbed by the sample and $T$ is the transmittance (the proportion of unabsorbed photons which pass through the sample). This gives the change in the number of MC-form molecules $\Delta c_{MC} = \Delta N_{MC} / N_A V$.

At the limit $\Delta t \rightarrow 0$, the concentration change of MC-form molecules is hence:

$$\frac{dc_{MC}}{dt} = -\frac{\Phi_d}{N_A V} \frac{I_d^0 \lambda_d A}{hc} (1 - e^{-\ln(10)\varepsilon_{MC}^{\lambda_d} c_{MC} l}) \qquad \text{Equation 8}$$

For samples with low OD (i.e. $\varepsilon_{MC}^{\lambda_d} c_{MC} l$ has a value ~0.1 or less), the incident photons are not appreciably attenuated as it passes through the sample. In this case, $e^x$ can be approximated as a truncated Taylor series expansion i.e. $e^x \approx 1 + x$ as $0.1^2 = 0.01$ which is negligible. This simplifies the concentration change equation to:

$$\frac{dc_{MC}}{dt} = -\frac{\Phi_d}{N_A} \frac{I_d^0 \lambda_d}{hc} \ln(10) \varepsilon_{MC}^{\lambda_d} c_{MC} l \qquad \text{Equation 9}$$

The solution to this differential equation is:

$$c_{MC}(t) = c_{MC}^0 e^{-k_1 t} \qquad \text{Equation 10}$$

where $k_1 = \frac{\Phi_d}{N_A} \frac{I_d^0 \lambda_d}{hc} \ln(10) \varepsilon_{MC}^{\lambda_d}$. Equation 9 describes the change in the concentration of MC molecules over time in response to illumination $I_d^0$. The concentration of SP molecules is then simply

$$c_{SP}(t) = c_{NB} - c_{MC}(t) \qquad \text{Equation 11}$$

as the total number of molecules in the sample is constant.

Similarly, the rate of colouration changes with the concentration of SP molecules. The SP-form molecules will absorb a fraction of the incident photons and transition to the MC form. In this case, the concentration of SP molecules is given by:

$$c_{SP}(t) = c_{SP}^{0} e^{-k_2 t} \qquad \text{Equation 12}$$

where $k_2 = \frac{\Phi_c}{N_A} \frac{I_c^0 \lambda_c}{hc} \ln(10) \varepsilon_{SP}^{\lambda_c}$ and the concentration of MC-form molecules is hence:

$$c_{MC}(t) = c_{NB} - c_{SP}(t) \qquad \text{Equation 13}$$

However it is not possible to directly measure the number of molecules in each form in a sample. Instead the intensity of fluorescence during decolouration is recorded. The emitted photons are collected by a lens with a solid angle ratio of $\Omega / 4\pi$, and directed to the photodiode by the optical system with an efficiency $\eta$, giving a measured intensity of

$$I_{MC^*} = I_d^{ab} \Phi_f \frac{\Omega}{4\pi} \eta \qquad \text{Equation 14}$$

where $I_{MC^*}$ is the measured intensity, $I_d^{ab}$ is the intensity of decolouration stimulus absorbed by the sample and $\Phi_f$ is the quantum yield of fluorescence.

The detected light is then converted into a current $I$ by the photodiode with responsivity $R$ (A/W) and amplified by the photodiode amp with gain G (V/A). This gives a measured signal of:

$$M_{MC^*} = RG\Phi_f \frac{\Omega}{4\pi} \eta I_d^0 \ln(10) \varepsilon_{MC}^{\lambda_d} c_{MC} l \qquad \text{Equation 15}$$

or

$$M_{MC^*} = k\, c_{MC} \qquad \text{Equation 16}$$

where $k$ is a constant.

As the intensity of fluorescence $I_{MC^*}$ is linearly proportional to the MC concentration $c_{MC}$, and as the measured voltage $M_{MC^*}$ is linearly proportional to $I_{MC^*}$, it is possible to determine the rates of colouration and decolouration while working

with the measured signal, without converting back to intensities or concentrations. The following modified equation for colouration is used when operating on measured voltages:

$$M_{MC*}(t) = M_{MC*}^{\max} - (M_{MC*}^{\max} e^{-k_2 t}) + M_{MC*}^{offset} \quad \text{Equation 17}$$

where $M_{offset}$ is a fixed increased in measured fluorescence, caused by experimental factors including background light, offset voltages in the electronics and the dark current of the photodiode.

Similarly, the modified equation for decolouration is:

$$M_{MC*}(t) = M_{MC*}^{0} e^{-k_1 t} + M_{MC*}^{offset} \quad \text{Equation 18}$$

Although some of the constants in Equation 15 are unknown, $k_1$ and $k_2$ can be measured experimentally and used to predict the rates of colouration and decolouration. Typical measured values for $k_1$ and $k_2$ are 4 $s^{-1}$ and 0.071 $s^{-1}$ respectively.

### 5.3. Thermal Relaxation

As discussed in *Chapter 2: Background Material*, a sample of NitroBIPS is subject to thermal transitions and will relax over time towards a temperature-dependant equilibrium. Thermal relaxation places a limit therefore, on the ability of registers to store data as stored values will change over time.

In this section the rate of thermal relaxation in eNBIPS samples, and its temperature dependence was investigated. A Peltier effect heating/cooling pad was attached to a heat sink, and used to heat or cool an eNBIPS sample. The temperature was recorded with a temperature probe. Samples of eNBIPS were then exposed to green or UV light to prepare the sample into a $M_{MC*}^{\min}$ or $M_{MC*}^{\max}$ state.

Figure 5.3 shows how an $M_{MC*}^{\min}$ eNBIPS sample thermally relaxes over time. Cooling the sample did not appear to change the rate of thermal relaxation, but did appear to improve $\Phi_{f,}$ causing a fixed increase in measured fluorescence. By comparison, heating the sample caused both a decrease in $\Phi_f$ and increased thermal transition rates. Because the SP form is more energetically favourable, the equilibrium state is dominated by SP molecules. Therefore thermal relaxation from

the more stable SP-dominated state is a minor effect; the measured fluorescence in Figure 5.3 was amplified 20x by the filter before any noticeable change could be found.

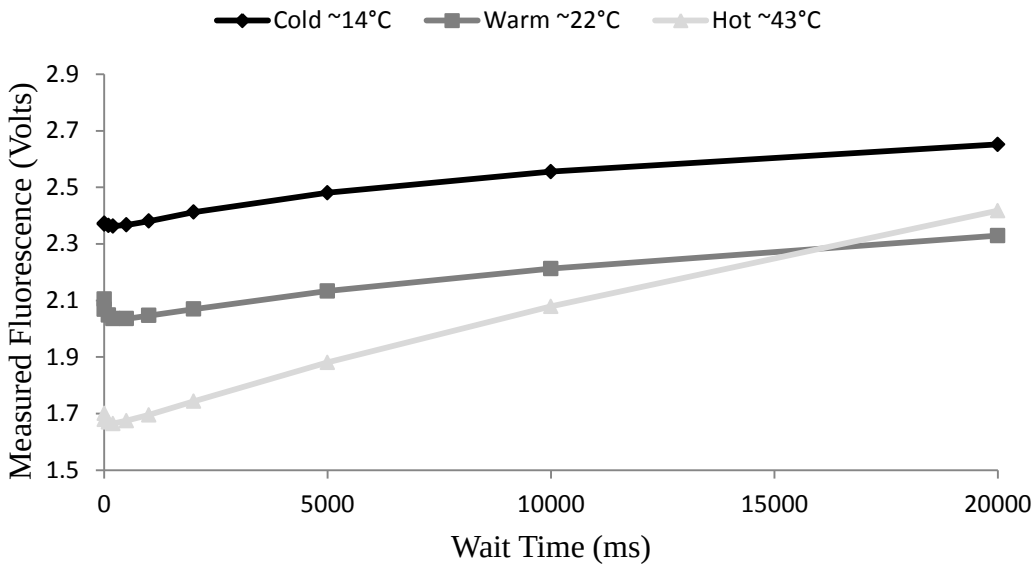



*Figure 5.3: Thermal relaxation over time for three temperatures of eNBIPS prepared to a* $M_{MC*}^{\min}$ *state by exposure to green stimulus.*

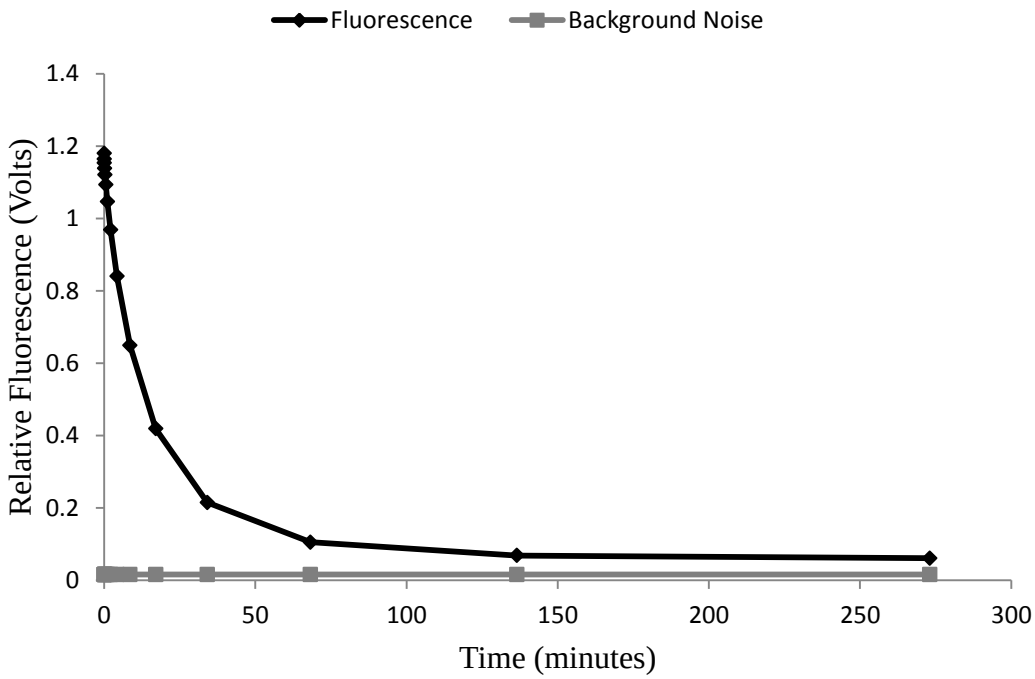



*Figure 5.4: Thermal relaxation over time for a sample of eNBIPS prepared to a* $M_{MC*}^{\max}$ *state by exposure to UV stimulus. Note that the final equilibrium reached is not equal to background noise, indicating the presence of some MC-form molecules.*

By comparison, when the eNBIPS was prepared to a $M_{MC*}^{\max}$ state by prolonged exposure to UV light, thermal relaxation was found to occur significantly faster as the state of the sample was further from equilibrium. Figure 5.4 shows a room-temperature eNBIPS sample colourised to $M_{MC*}^{\max}$ and left for a time $T$ before the fluorescence was measured. This ensures that the rate of thermal relaxation is not affected by the green measurement stimulus. The measured decay constant is 0.0012 $s^{-1}$; approximately 3000 times slower than decolouration. However, with a half-life $t_{1/2}$ of 583 s, a sample left for just ten minutes will have decayed to half its original value.

## 5.4. Overview of Photochromic Registers

As discussed in *5.2*, in a sample of eNBIPS the total concentration of molecules $c_{NB}$ is composed of several sub-populations of molecules. These are the

SP-form molecules ($c_{SP}$) and the MC-form molecules ($c_{MC}$), such that $c_{NB} = c_{SP} + c_{MC}$.

As discussed in 3.1, a sample of molecular switches can store data as the relative concentrations of molecules in each of the two stable states, giving a non-binary integer register. An integer register stores a value, from $V_0$ to $V_{max}$, where max is the maximum capacity of the register. Any switching molecule can have data encoded in this manner.

We designate SP as the base form, as SP is the more stable of the two forms. A sample consisting of $c_{SP}^{\max} + c_{MC}^{\min} = V_0$, and $c_{SP}^{\min} + c_{MC}^{\max} = V_{max}$. Due to thermal transitions, $V_0$ will not consist solely of SP-form molecules, and $V_{max}$ will not consist solely of MC-form molecules. Thermal transitions prevent us setting a sample to a homogeneous state.

The maximum capacity of the register is determined by how many distinct concentrations $c_{MC}$ can be reliably distinguished. As only MC-form molecules fluoresce when exposed to visible light, the intensity of the fluorescence is proportional to $c_{MC}$. As a result, the capacity of the register is the number of levels of measured voltage $M_{MC*}$ which can be reliable distinguished when the register is subjected to excitation stimulus. Each distinguishable voltage is assigned a ‘value’; the number that register represents.

Altering the value of the register is via the application of light. Colouration stimulus increments the register by increasing $c_{MC}$, and decolouration stimulus decrements it by decreasing $c_{MC}$. This causes a corresponding change in $M_{MC*}$ when the register is subject to excitation stimulus. The duration of pulses should be determined, but we can abstract with the representation shown in Chapter 3:, where $\uparrow_n$ represents a colouration of $n$ values, and $\downarrow_n$ is a decolouration of $n$ values, subject to the equations:

| | | |
|---|---|---|
| $V_x + \uparrow_y = V_{x+y}$ | $x + y < max$ | Equation 1 |
| $V_x + \uparrow_y = V_{max}$ | $x + y \geq max$ | Equation 2 |
| $V_x + \downarrow_y = V_{x-y}$ | $x > y$ | Equation 3 |
| $V_x + \downarrow_y = V_0$ | $x \leq y$ | Equation 4 |

Reading the register requires a pulse of excitation stimulus. However, for NitroBIPS an excitation stimulus and a decolouration stimulus are the same. As a result any register read will also be decoloured by the stimulus. As a further complication, a decolouration pulse of fixed duration will cause more or less decolouration depending on the state of the sample. The higher the concentration of MC molecules, the faster the sample decolours (as the probability of a photon being absorbed by a MC-form molecule as opposed to an SP-form molecule is higher). To compound this problem, the nature of registers means the state of the sample is not known in advance; that is the purpose of the read pulse!

There are multiple strategies to deal with this issue. The first is short-pulse reading. Short-pulse reading uses read pulses of a very short duration such that the level of decolouration caused to the register is minimal. Short-pulse reading is the simplest read method but the short pulse duration means measured fluorescence is highly prone to noise and decolouration will add up over time, leaving the register in an undefined state.

The second strategy is full reset reading. This requires any read pulse to completely decolour a register to $M_{MC*}^{\min}$, and then write the register back to its previous value. This destructive read process is akin to magnetic core memory [Forrester (1951)], always leaving the register in $V_0$ at the end of the read operation. This method provides a large number of data points for measuring fluorescence making it robust to noise. It also eliminates the change in register state due to the decolouration of the read pulse. However, this method is the most time consuming, both due to the long duration of the decolouration pulse, but also the optional requirement to re-set to register.

The third strategy is decrement reading. Decrement reading is the most complex read method and is the method we use for eNBIPS register implementation. A read pulse is a fixed duration such that each read decrements the register by exactly one value. As the rate of decolouration is non-linear, the values are distributed non-linearly along the dynamic range of fluorescence. A possible modification to decrement reading is decrement reading with resetting; where the register is reset back to its previous state after each read. The following section discusses how decrement reading is performed.

## 5.5. Photochromic Register Implementation

### *5.5.1 Initialisation*

A photochromic register before use is in an unknown state, and with unknown rates of colouration and decolouration. For this implementation a photochromic register must be initialised before use. This is required once per execution (not per sample) as bleaching will cause a decrease in the maximum level of fluorescence over time and changes to ambient temperature can cause the alteration of the rate of thermal relaxation and a change to the fluorescence offset $M_{MC*}^{offset}$. For very long executions, re-initialisation may be necessary at periodic intervals as the sample bleaches. Initialisation involves several steps:

1. Determine the range of fluorescence.
2. Define the usable range.
3. Determine the signal to noise ratio.
4. Calculating the read time.
5. Determine the value fluorescence mid-points.
6. Calculating the value fluorescence bounds.
7. Calculating the value transition table.

The LabVIEW implementation of any system that implements registers or any components which are extensions of registers must complete these steps before use. A LabVIEW state machine pattern is used to execute each step in turn, and save the read pulse duration, value mid points, bounds and transition table to LabVIEW data clusters, which are used during the rest of execution.

***Determine Range of Fluorescence***

Step one is to determine the range of fluorescence. This allows us to place register readings on the scale from minimum to maximum. To achieve this, the sample is decoloured to $M_{MC*}^{\min}$ using an arbitrarily long pulse of decolouration stimulus. The register is then exposed to excitation stimulus, and the fluorescence measurement is taken. This is the minimum possible fluorescence $M_{MC*}^{offset}$ from Equation 18 and will be non-zero due to background signal, the fluorescence from MC-form molecules created by thermal transitions and imperfections in the dichroic optics.

Secondly, the register is colourised to $M_{MC*}^{\max}$ using an arbitrarily long pulse of colouration stimulus. The register is then exposed to excitation stimulus and the fluorescence measurement recorded. As the excitation stimulus will also decolour the sample, the measurement is taken during a short time period so the decolouration is negligible. We typically use the average of the first 5ms of stimulus (after LED turn-on delay). This gives $M_{MC*}^{\max}$ from Equation 17. The range of fluorescence $M_{range}$ is then $M_{range} = M_{MC*}^{\max} - M_{MC*}^{offset}$

During this process, a complete trace of colouration from $M_{MC*}^{offset}$ to $M_{MC*}^{\max}$ is recorded (by interspersing the arbitrarily long colouration pulse with brief read pulses), and a complete decolouration trace $M_{MC*}^{\max}$ from $M_{MC*}^{offset}$ to is also recorded (during the measurement for $M_{MC*}^{\max}$). By fitting the colouration/decolouration formulae Equation 17 and Equation 18 to these voltage traces, values for $k_1$ and $k_2$ can be estimated. This then allows a function describing the time to transition between any two levels of fluorescence to be determined. LabVIEW features built-in curve fitting sub-VIs to achieve this. The Nonlinear Curve Fit VI is used which implements the Levenberg-Marquardt fitting algorithm [Levenberg (1944)].

***Definition of Usable Range.***

As shown by section *5.2 Optical Theory*, the rates of colouration and decolouration are non-linear. The bottom of the range is more sensitive to colouration stimulus, and the top of the range more sensitive to decolouration stimulus. Registers in these states are more prone to the effects of stray light or irregularities in pulse duration. Additionally, the further away from thermal equilibrium the register is, the quicker thermal effects will change the state of the register.

To alleviate this, the range is restricted and only the middle region is considered for value midpoint placement. This is referred to as the usable range. The restriction can be user selected. Our experiments were conducted with the bottom third and top sixth of the fluorescence range excluded, which was found to give the most reliable results without compromising too heavily on the number of values. As an additional benefit to this strategy the distribution of value mid-points will be more linear.

$$V_0 = M_{MC*}^{offset} + \underline{r}M_{range}$$ Equation 19

$$V_{\max} = M_{MC*}^{\max} - \bar{r}M_{range}$$ Equation 20

where $\underline{r}$ is the restriction to the bottom of the range, and $\bar{r}$ is the restriction to the top of the range.

***Determination of Signal to Noise***

The noise in the system inhibits the ability to determine the level of fluorescence reliably. As background noise is normally distributed, it can be measured and a standard deviation σ calculated. The reliability of measurements of a register is then chosen as the proportion of measurements *p* that fell within *z* standard deviations of the expected value:

$$p = erf\left(\frac{z}{\sqrt{2}}\right)$$ Equation 21

where *erf* is the error function:

$$erf(z) = \frac{2}{\sqrt{\pi}}\int_0^z e^{-x^2}dx$$ Equation 22

For example, 68.3% of measurements will be within 1σ (either side) of the expected value, and 99.7% of measurements will be within 3σ.

Assuming a desired >99% reliability, then each distinguishable mid-point must be separated by >6σ (3σ either side of each mid-point), shown in Figure 5.5. As the distribution of value mid-points is not linear, this restriction must be valid for $V_0$; the value with the smallest distance to the next value.

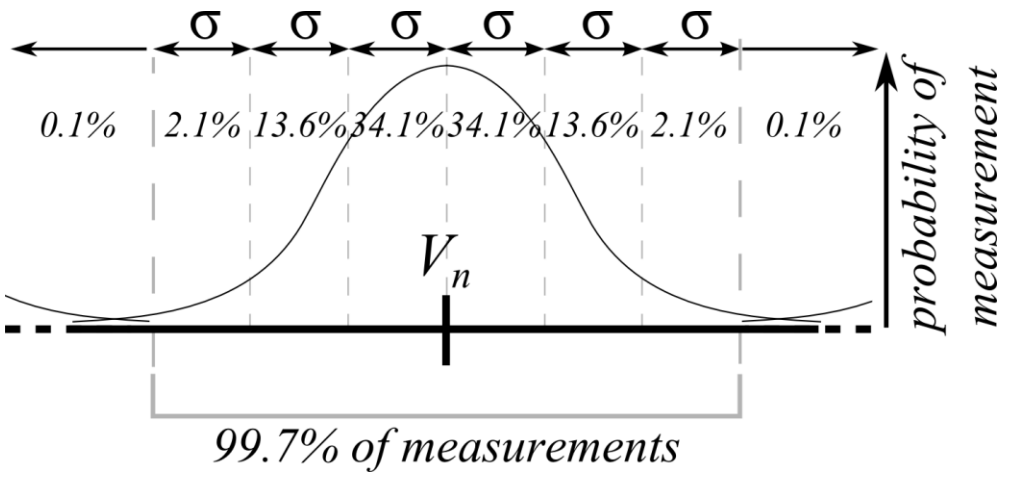



*Figure 5.5: Depiction of a single value and the distribution of recorded measurements due to system noise.*

As an example, the standard deviation of noise in a typical experiment was σ = 0.0005. With a usable range from 1.16 V to 0.49 V, this corresponds to 447 values separated by 3σ (18 if they're distributed non-linearly). By comparison, with the Benchmaster filter turned off, the standard deviation σ = 0.0127, or 17 values separated by 3σ (8 non-linearly).

***Calculating Read Time***

The read time is the duration of excitation/decolouration stimulus that both causes fluorescence to determine the value of the register (before the stimulus), and decolours by a single value, so the register remains in a known value after the read pulse.

For a register with desired capacity *n*, the duration of the read pulse $t_{rp}$ is the time required to decolour from $V_{max}$ to $V_0$, divided by *n-1*:

$$t_{rp} = \frac{t_{(V_{\max} \xrightarrow{decolour} V_0)}}{n-1}$$ Equation 23

The value for $t_{(V_{\max} \xrightarrow{decolour} V_0)}$ can be determined from the fitted decolouration formula.

***Determining Value Fluorescence Midpoints***

Each value $V_n$ has three fluorescence levels associated with it; the maximum value which is still interpreted as $V_n$ rather than $V_{n+1}$ (Upper bound, $\overline{V_n}$), the

minimum value which is still interpreted as $V_n$ rather than $V_{n-1}$ (Lower bound, $\underline{V_n}$) and the canonical value for $V_n$ which is aimed for when changing value (Midpoint, $\langle V_n \rangle$).

$\langle V_{\max} \rangle$ is the maximum fluorescence in the usable range. $\langle V_0 \rangle$ is the minimum fluorescence in the usable range. The midpoint of a value is the next highest value decoloured by a read pulse $t_{rp}$. Either the decolouration fit can be used to determine the midpoints, or they can be determined experimentally (which accounts for experimental error e.g. LED turn-on delay). The distribution of midpoints can be seen in Figure 5.6, and an example from our experiments can be seen in Table 3.

As a register could be read while in $V_0$, the read pulse will cause decolouration to $V_{-1}$. A midpoint for $V_{-1}$ must also be determined, to be used in calculating the time to colour from $V_{-1}$ to $V_0$.

| **Value** | $V_{-1}$ | $V_0$ | $V_1$ | $V_2$ |
|---|---|---|---|---|
| **Fluorescence** | 0.57 | 0.86 | 1.46 | 2.76 |

*Table 3: Example value midpoints as calculated for a ternary register.*

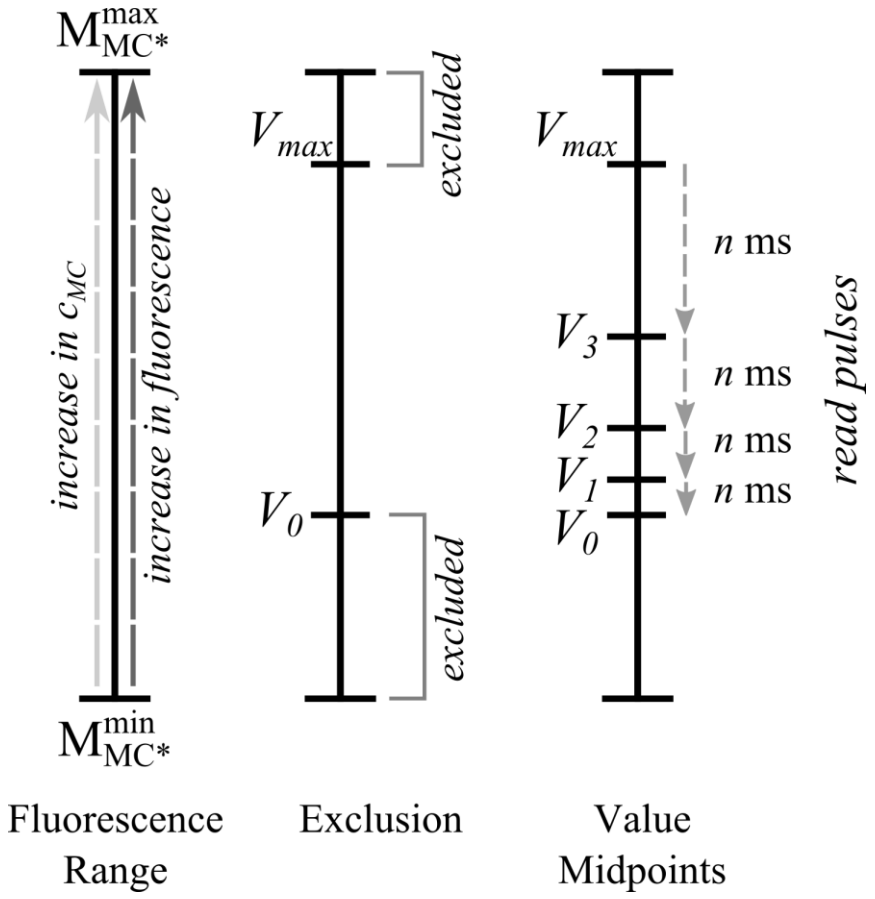



*Figure 5.6: How the fluorescence range is restricted and midpoints distributed.*

***Calculating Value Fluorescence Bounds***

The fluorescence bounds are the extremities of a value, the maximum and minimum recorded fluorescence that is still interpreted as that value. The upper bound for $V_{max}$ ($\overline{V_{max}}$) is $+\infty$. The lower bound for $V_0$ ($\underline{V_0}$) is 0. For all other bounds, $\overline{V_n}$ lies between $\langle V_n \rangle$ and $\langle V_{n+1} \rangle$, and $\underline{V_n}$ lies between $\langle V_n \rangle$ and $\langle V_{n-1} \rangle$. $V_{-1}$ does not have bounds, as it should never be read.

Due to thermal relaxation, the state of a sample will decolour over time. To alleviate this, we skew our bounds, such that $\overline{V_n} = \langle V_n \rangle + \bar{s}(\langle V_{n+1} \rangle - \langle V_n \rangle)$ and $\underline{V_n} = \langle V_n \rangle - \underline{s}(\langle V_n \rangle - \langle V_{n-1} \rangle)$, where $\bar{s}$ and $\underline{s}$ are skew factors, provided $\overline{V_n} - \langle V_n \rangle$ and $\langle V_n \rangle - \underline{V_n} \geq 3\sigma$ of noise. We found using $\bar{s} = \frac{1}{6}$ and $\underline{s} = \frac{5}{6}$ provided the best balance between offsetting thermal relaxation, while still accounting for measurement noise.

All the value bounds (and midpoints) are saved to the value bound table (VBT), and example of which is Table 4. The lower bound of $V_n$ is equal to the upper bound of $V_{n-1}$, shown above and below the scale in Figure 5.7.

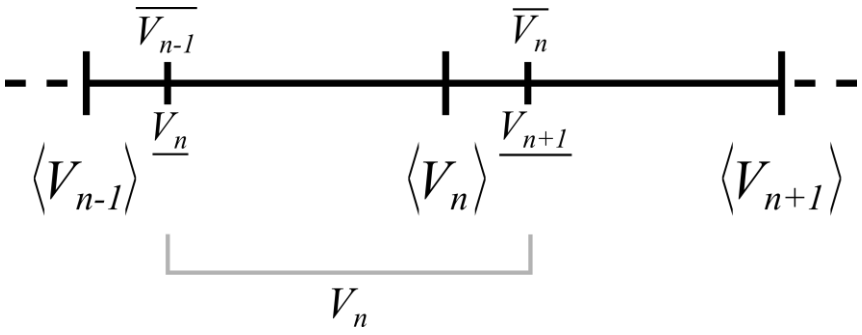



*Figure 5.7: Distribution of value midpoints and bounds. Lower bounds are further from the midpoints than upper bounds.*

| **Value** | $V_{-1}$ | $V_0$ | $V_1$ | $V_2$ |
|---|---|---|---|---|
| **Upper Bound** | N/A | 1.16 | 2.11 | $+\infty$ |
| **Midpoint** | 0.57 | 0.86 | 1.46 | 2.76 |
| **Lower Bound** | N/A | 0 | 1.16 | 2.10 |

*Table 4: Example upper and lower bounds as calculated for a ternary register.*

***Calculate Value Transition Table***

The value transition table (VTT) is a two dimensional square table. One axis is the current register value, and the other is the desired register value, and the corresponding cell is the duration of stimulus necessary to colour/decolour to that value's midpoint. An example from our experiments is Table 5.

| **To →**<br>**From ↓** | $V_0$ | $V_1$ | $V_2$ |
|---|---|---|---|
| $V_0$ | 1771 | 6067 | 22,746 |
| $V_1$ | 0 | 4296 | 20,975 |
| $V_2$ | -168 | 0 | 16,679 |

*Table 5: Example Value Transition Table for a ternary register.*

The colouration entries in the VTT can be calculated using the fitted equations Equation 17 and Equation 18. The decolouration entries (demarcated as

being negative numbers) will be multiples of the read pulse duration. Note that the VTT considers the current value to be the value read by the read pulse, not the value the register was left in due to the decolouration of the read pulse. Hence the stimulus required to transition from $V_n$ to $V_n$ is not zero, but the stimulus needed to transition from $V_n$ to $V_{n-1}$ is zero.

At the conclusion of VTT generation, the register is set to $V_0$ and is ready for use. The VTT, VBT and read pulse duration are all stored by LabVIEW and can be used by subVIs later in execution.

### *5.5.2 Reading*

Reading a register is implemented by exposing the sample to a read pulse with the duration calculated during the initialisation. This causes the sample to fluoresce, which is recorded by the photodiode and converted into a voltage. The recorded voltage is then compared to the set of value bounds to determine which value corresponds to the reading.

The level of fluorescence is recorded continuously for the duration of the read pulse. The recorded fluorescence forms an exponential decay function. As only a single value is required to correspond to the VBT, the level of fluorescence at the beginning of the read pulse is calculated. There are two methods for this:

The first is to use a function fitting method to fit the recorded fluorescence to the decolouration formula, as shown in Figure 5.8 and Figure 5.9. This then gives a value for $M^0_{MC^*}$. This method is robust to noise, and also robust to LED turn on time. However, as the recorded level of fluorescence is sometimes almost flat (when reading $V_0$), the fitting method can return wildly incorrect values for $M^0_{MC^*}$. An exponential decay function is of the form $y = e^{(-kt)}$ and the constant $k$ determines how rapidly the function decays to a flat state. If the recorded data is completely flat, the fitting algorithm will fit a function with a very high value for $k$, with a corresponding very high (and wildly inaccurate) value for $M^0_{MC^*}$.

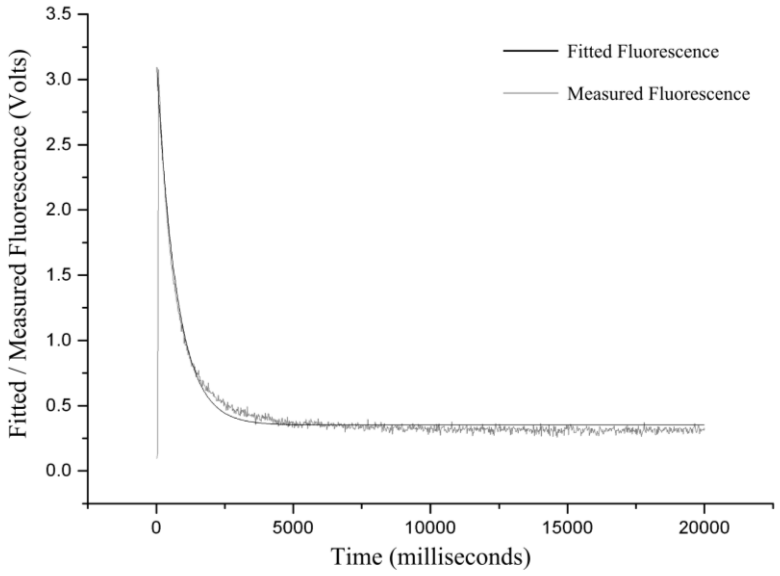



*Figure 5.8: Comparison of measured fluorescence and fit for a fully decoloured register.*

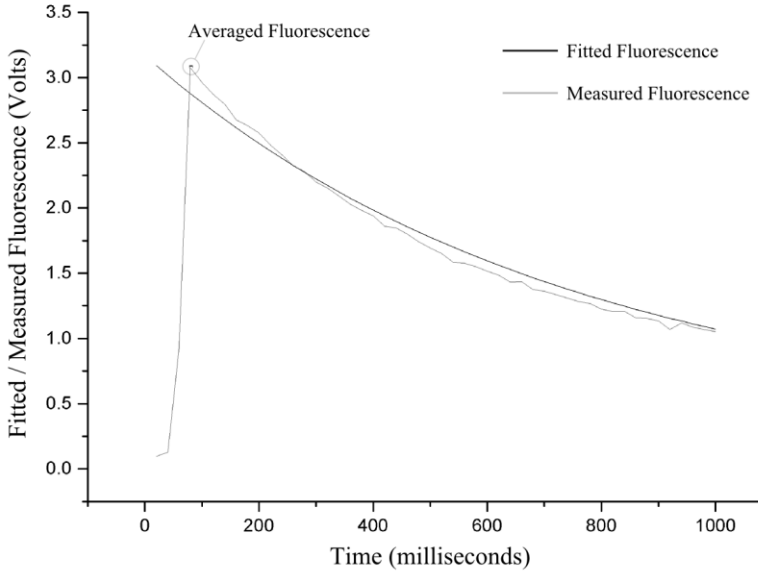



*Figure 5.9: Zoomed view of the first second of Figure 5.8, showing how the peak measured fluorescence and peak fitted fluorescence compare. The fitted fluorescence peak was 3.09, and the 5ms average was 3.07.*

The second alternative is to discard the recorded fluorescence during the LED turn on time, and average the first readings for a short period of time. For example, we average the first 5ms of fluorescence (typically 5 measurements) to produce a value. Longer durations can be used to increase robustness to noise, at the cost of returning a value that is lower than the actual level of fluorescence. This method works even if the fluorescence decay is flat.

The read operation decrements the register by one and the VTT accounts for this. However, unless the system implements decrement reading with resetting, the register cannot be read twice in a row without a write operation occurring first. Doing so would decrement the register twice, potentially taking it below $V_{-1}$, where transition times are no longer defined. Decrement reading with resetting can be used to remove this limitation, where every read operation is immediately followed by a $\uparrow_1$ write operation, returning the register to the value from before the read.

An additional consideration when reading the register is that the register is subject to thermal relaxation over time and the state of the sample will change (typically a decrease towards thermal equilibrium). As a result, the register will eventually transition to an unknown state. To combat this, the read pulse calculates a 5ms moving average of the level of fluorescence. If this average falls below the midpoint of the value the register is transitioning to, the read pulse immediately ceases, preventing overshoot and returning the register to a known state.

Reading a register used a subVI extended from the Decolour LabVIEW subVI described in *Chapter 4: Methods*. The extended Read sub-VI takes the VBT and an averaging window time as additional inputs, and outputs both the value the register was determined to be in, and the actual recorded level of fluorescence (for data logging purposes), shown in Figure 5.10. The subVI will immediately calculate the average fluorescence at the start of the read pulse by ignoring the LED turn on period, then averaging the fluorescence for the averaging time and compare the result to the VBT to determine the value $V_n$. Once this is done, the subVI also loads $\langle V_{n-1} \rangle$ and continues to calculate a moving window average. Should this average dip below $\langle V_{n-1} \rangle$, the decolouration immediately ceases, seen in Figure 5.11.

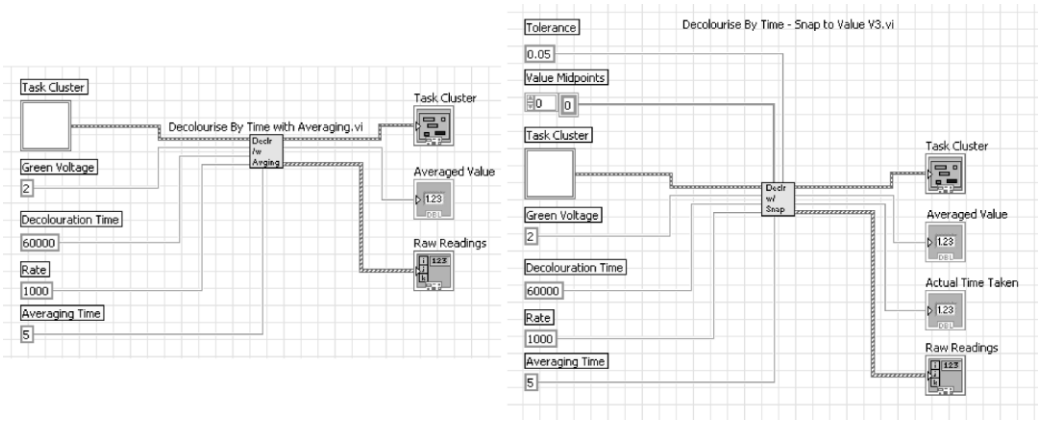



*Figure 5.10:* `Decolourise by Time with Averaging.vi` *and* `Decolourise By Time - Snap to Value V3.vi`*. Both subVIs are extensions to the basic decolour subVI. The former calculates the peak fluorescence by averaging for a defined time period. The latter finds the peak fluorescence, and also ceases the decolouration early if the fluorescence crosses a value midpoint.*

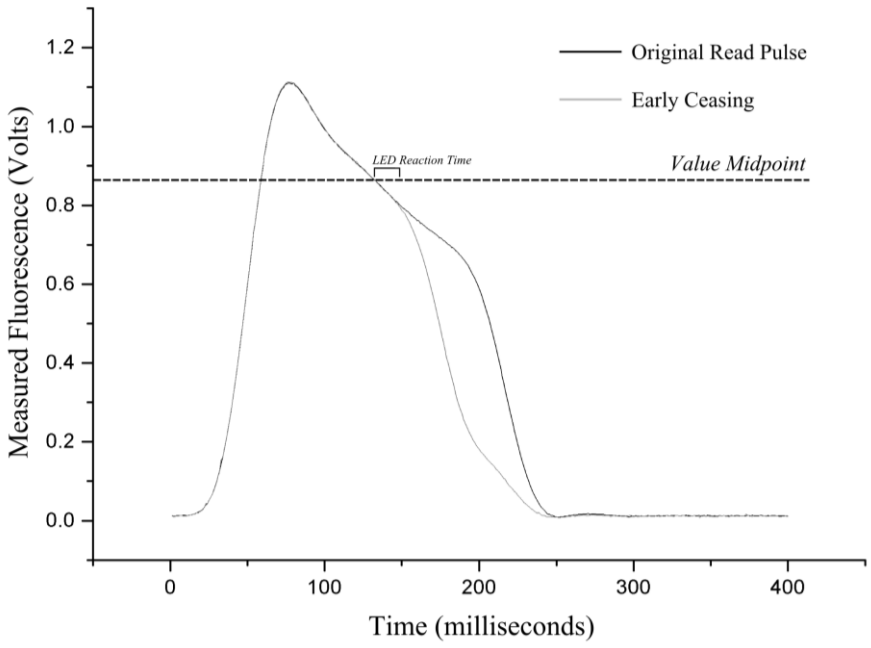



*Figure 5.11: A register written to* $V_1$ *decolouring to* $V_0$ *with and without early ceasing of stimulus. As thermal relaxation has caused the sample to partially decolour, the original read pulse will overshoot the desired value midpoint (though as* $V_0$*, it will remain within the bounds). Note however, that there is still a delay between fluorescence crossing the threshold and the LED beginning to turn off.*

An alternative read subVI implements function fitting to determine the peak fluorescence, but has two disadvantages. Firstly, the fitting must be concluded at the end of the read pulse so it cannot cease the pulse early to prevent overshoot. Secondly, reading a low value (typically $V_0$) will cause the fitting algorithm to return an incorrect result as discussed earlier.

### *5.5.3 Writing*

Transitioning a register between values is via the application of colouration or decolouration stimulus in accordance with the VTT. Writing requires knowledge of the state of the register so a read operation must be performed first. The recorded value and the desired value are cross-referenced on the VTT to give the amount of colouration or decolouration stimulus required to transition the register to the new value.

Stimulation with colouration stimulus will increment the register. Incrementation is blind; there is no feedback as to the progress of the operation in the form of fluorescence. Colouration will trend towards slightly undershooting (The peak fluorescence for $V_1$ and $V_2$ in Figure 5.12 show this), as the LED takes time to reach full intensity, and this delay is not included in the fitted colouration equation which determines the duration of the pulse. This is another reason why the bounds of values are skewed. Incrementation is implemented with an unmodified Colour subVI, as described in *Chapter 4: Methods*. The colour subVI is given the appropriate pulse duration from the VTT.

Stimulation with decolouration stimulus will decrement the register. Decrementation is not blind; the system records feedback as fluorescence allowing the system to track the progress of the operation. As mentioned with reading, the state of the sample trends to decrease over time due to thermal relaxation. The feedback during decolouration allows us to take a moving average, and cease the decrementation early if the operation would decolour too far. This prevents thermal relaxation from causing incorrect readings, provided the register is used frequently. Decrementation is implemented with the modified Decolour subVI. It is given both the appropriate pulse duration from the VTT, but also the midpoint of the target value and an averaging time so implements the same early ceasing of decolouration as described in *5.5.2*.

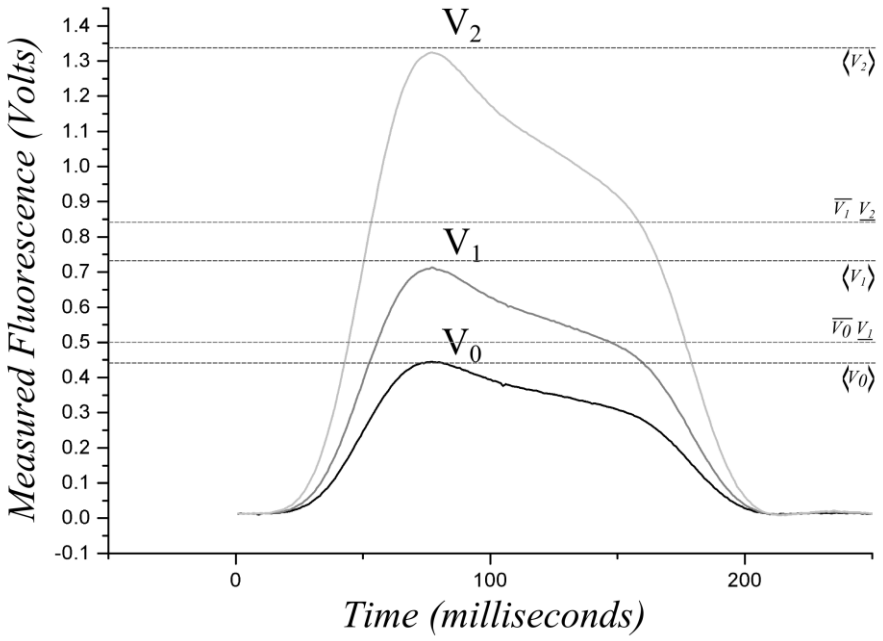



*Figure 5.12: An example ternary register stepped upwards through states.*

### *5.5.4 Parallel Registers*

Sub-regions of eNBIPS can possess different concentrations of MC-form molecules. By addressing different regions of an eNBIPS sample, it is possible to store more than one value on a sample. This allows the storage of larger numbers by treating each *n*-capacity register as a base-*n* digit in a number.

Use of parallel registers requires an additional step during register initialisation:

***Determine Motor Step Size***

The motor has a limited travel $\Delta$ (10mm for our setup). Illumination has a fixed width $w$ (typically 0.4mm for our setup). The user must input these two values, and the system will calculate the maximum number of parallel registers $p$ and the step size $\Delta_s$; the distance the motor must move to transition to the middle of the sub region that stores a register.

$$p = \lfloor (\Delta / w) + 1 \rfloor \qquad \text{Equation 24}$$

$$\Delta_s = \frac{\Delta}{(p-1)}$$ Equation 25

As eNBIPS is homogenous, the initialisation procedure needs only be executed on a single sub-region, and the results will be applicable to all parallel registers in the system. Parallel registers are treated exactly like regular registers but the register being addressed can be changed by using a Move subVI, as described in the methods section.

An example of a parallel register being incremented is shown in Figure 5.13. A four-element parallel register counter counts in ternary from 0 to 80 ($3^4$ = 81 possible values), as executed on our system. A typical eNBIPS sample is used as described in 4.1: eNBIPS Samples. Illumination times are as shown in Table 5. Table 6 is the VBT for these parallel registers. The figures given were automatically determined by the register initialisation procedure.

Each iteration is labelled with the decimal representation, the state of each register ($V_0$: White, $V_1$: Grey, $V_2$: Black) and the fluorescence readings recorded during execution. As sub-regions are read from right to left, the fluorescence readings for the most-significant bit (furthest left) is typically lower as the register is subject to thermal relaxation. This is most obvious during iterations 0 and 80, where the measured fluorescence decreases from right to left.

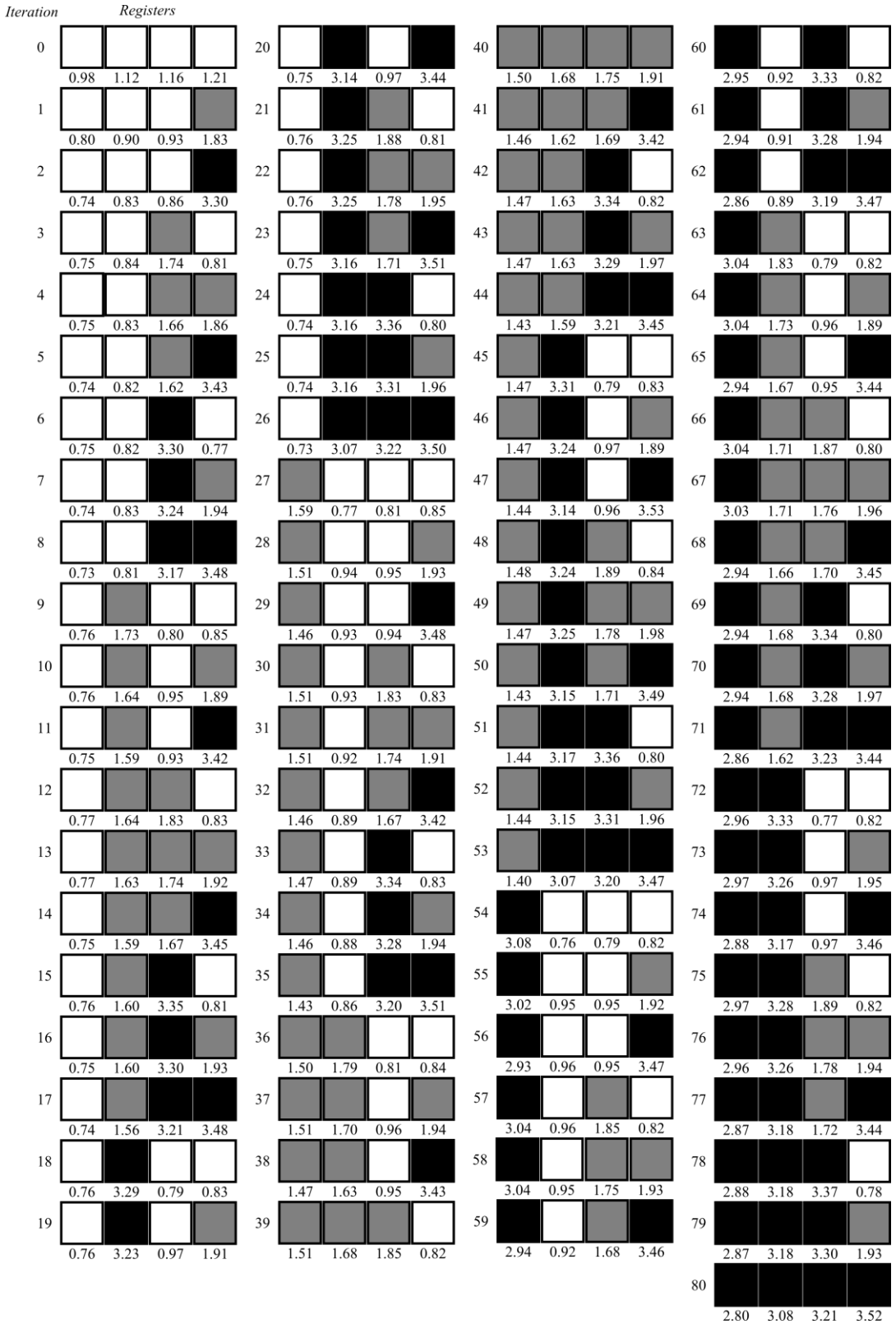



*Figure 5.13: A summary of a four-element parallel register counter counting in ternary from 0 to 80 .*

| **Value** | $V_0$ | $V_1$ | $V_2$ |
|---|---|---|---|
| **Upper Bound** | 1.31 | 2.41 | $+\infty$ |
| **Midpoint** | 0.96 | 1.66 | 3.14 |
| **Lower Bound** | 0 | 1.31 | 2.41 |

*Table 6: Value table for the ternary registers in Figure 5.13.*

Repeatability is an issue, even within a simple parallel register. For example, recorded fluorescence from cell 0 storing $V_2$ in Figure 5.13 varies from 2.80 to 3.08, decreasing as the time required to illuminate the remaining cells increase. This decrease in stored value is due to the thermal relaxation of the sample over time, and is more pronounced at higher MC concentrations, making higher values less stable. It is this issue that limits our current implementation of parallel registers to 4-register ternary; higher numbers of registers caused a larger time delay between a register being written and read, and higher bases reduce the time a register can thermally relax before it is erroneously read as a lower value.

## 5.6. Discussion

In this section we have demonstrated the implementation of photochromic registers capable of storing data as the relative concentration of MC-form molecules. Registers are written to via the application of colouration or decolouration stimulus, and read via excitation stimulus-induced fluorescence. The examples shown demonstrate how we can implement registers, both singularly and in parallel. However, parallel registers using our current experimental architecture cannot be written to or read in parallel, though this could be implemented with the use of a spatial light modulator. Interestingly, three dimensional addressing of sub-regions could be implemented using two-photon excitation. Both these extensions are discussed in *Chapter 9: Discussion and Conclusions*.

Photochromic registers differ from conventional data storage methods in that reading the register will also change the state, though not necessarily the value if the read pulse is sufficiently short. As excitation stimulus and decolouration stimulus are the same, inducing fluorescence will also induce decolouration. We described methods to counter this, by distributing values such that any read operation

decrements the register by one, leaving it a known state. The register could then be used as is, or have its state reset with an increment operation.

Two factors limit our ability to store more values with each register. The first is noise, and this reduces the number of distinguishable fluorescence levels the photodiode can determine. The hardware filter and averaging or function fitting partially alleviates this, but it remains an issue. The second is thermal relaxation, causing the concentration of MC-form molecules to relax to thermal equilibrium. Photochromic registers are hence more applicable to short-term data storage, storage of continuous values or storage where precision is less critical. A system that interacts with photochromic registers could distribute more values closer together at the expense of increased likelihood of reading a neighbouring value. This could go as far as having a continuous registers, where the value and the fluorescence reading correspond 1:1. Conversely, it could store fewer values with increased reliability. This change could be made with no alteration to the eNBIPS sample itself.

Register failure is typically the result of storing a high value for long periods, causing the sample to thermally relax to a lower value, which is subsequently read. Higher values are considerably less stable than lower values, as there are more MC-form molecules to potentially relax. This type of failure is particularly troublesome as there is no indication that the reading is incorrect. Preventing failure requires a large difference between two values at the higher end of the range, limiting the potential capacity of registers. Alternative failure modes are due to noise in the system causing an incorrect fluorescence measurement, or incorrect pulse durations based on faulty assumptions of the register, such as its current state that has changed due to thermal relaxation.

Improving the capacity of photochromic registers depends on decreasing the effects of noise and decreasing the rate of thermal relaxation. Noise in our experimental system is from many sources, including electromagnetic noise and scattered light. Any reduction in EM noise or scattered light would allow the discrimination of more values. The limit to this is shot noise, the noise inherent to fluorescent processes; small amounts of fluorescence have a high variability in the amount of emitted photons per unit time. Thermal relaxation is inherent to NBIPS itself and the temperature of the system. The alternative is to use a species of

spiropyran with lower quantum yields or use a solvent that inhibits molecular mobility. This would increase the time required to alter the state of the register, but decrease the rate of thermal relaxation.

# Chapter 6: Steps Toward Photochromic Molecular Turing Machines

As discussed in *Chapter 2: Background Material*, the Turing machine is Alan Turing's theoretical description of a minimal computing device that could execute any algorithm [Turing (1936)] [Boolos et al. (2002)]. A Turing machine is the quintessential model of universal computation, and this chapter describes how we can utilise eNBIPS registers to implement the tape in a Turing machine as an example application of eNBIPS registers.

## 6.1. Implementation of Photochromic Tape Turing Machines

The definition of a Turing machine consists of four primary components (as shown in Figure 6.1) and five fundamental actions. Each component requires a corresponding implementation in our hardware setup, and each action operates on these components. The four primary components are:

1. *Tape*. A Turing machine tape is a bidirectional limitless one-dimensional array of cells. Each cell stores a letter from a defined alphabet Γ, defaulting to a specified blank value. This is represented as a sample of eNBIPS with parallel registers. Each cell is a register along the sample, where the register's

capacity $d \geq |\Gamma|$. If $d < |\Gamma|$, then *k* registers, each with capacity *d*, could be used as a virtual cell with capacity $d^k \geq |\Gamma|$.

In this pilot study the illumination size is 0.4 mm and the stepper motor has a travel of 10 mm, giving a maximum tape length of 25 cells. This limits the possible algorithms that can be executed but is sufficient for a proof of principle. Photochromic Turing machine executions begin in the centre of the tape and aborts execution should the tape bounds be exceeded.

2. *Head*. A Turing machine head reads and writes to a cell on the tape. This is implemented by illumination and detection hardware addressing a register on the eNBIPS tape. The stepper motor allows the head to move.
3. *I-State Register*. An internal state register that stores the Turing machine's current state from a set of possible internal states *Q*, including a defined start I-state, and a halt I-state. Two options exist for implementation. Early efforts stored the I-State on the controlling computer and updated it in response to the *update* action. The alternative is to store the I-state on the tape itself. We allocate the left-most cell (as the tape is limited) or left most cells (if the register capacity is smaller than |*Q*|) as the I-State register, seen in Figure 6.2. This I-State can then be read and updated with modified *update* and *decide* actions. For simplicity, we retain the same base as the rest of the Turing Machine. An *n*-state *m*-symbol machine would require $\lfloor \log_m n \rfloor + 1$ I-state cells.
4. *Transition Function*. The transition function δ is the instruction set for the Turing machine. It is commonly rendered as a two dimensional table defining what actions to perform depending on the value of the read cell and the I-state. The actions are a 3-tuple, with instructions to *print, move* and *update*. We store the transition function on the controlling computer, but we later discuss how in principle it could be stored on the tape.

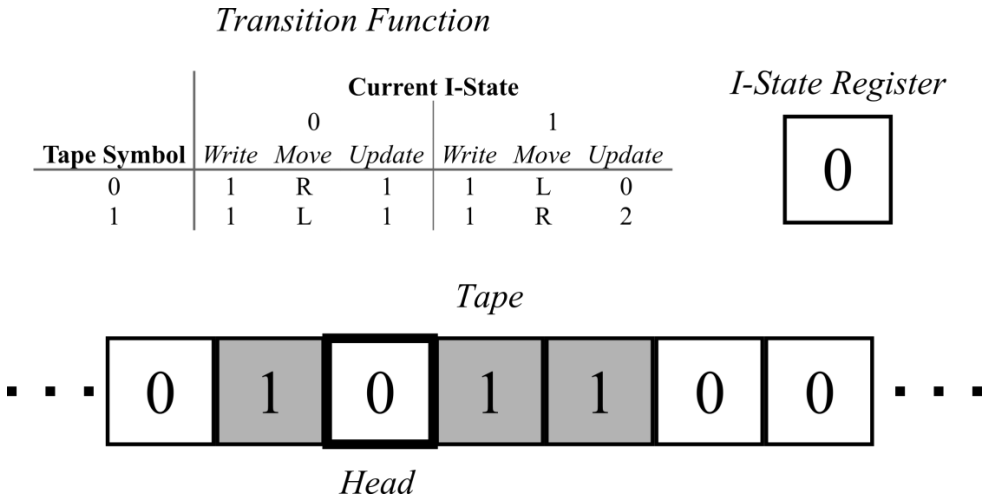



*Figure 6.1: The four components of a classical Turing machine, as previously seen in Figure 2.1.*

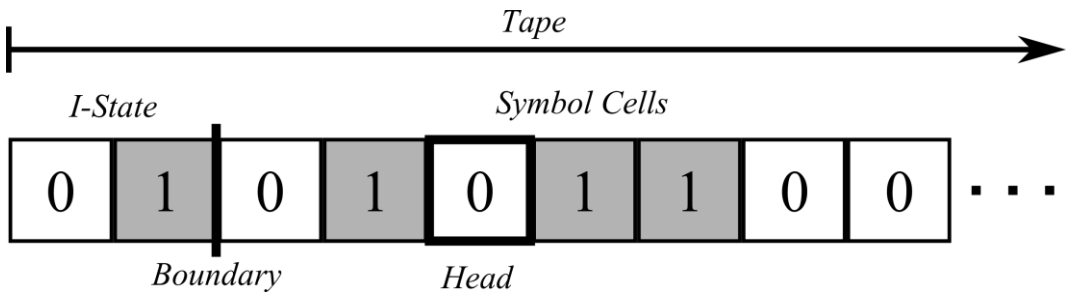



*Figure 6.2: The layout of the photochromic Turing machine tape. In this example the I-state is saved on the tape in a pair of binary registers allowing up to four I-states.*

In addition to these components, the system must enact five fundamental actions; *Observe*, *Decide*, *Print*, *Move* and *Update*. The nature of photochromic registers requires two additional compound actions; *Initialise*, that prepares the photochromic registers representing the tape for use, and *Output*, which reads the values of registers on the cell, producing the system output.

1. *Initialise*. Operates on: Tape, Head. The registers implementing cells in a photochromic Turing machine tape require initialisation before use, as described in *Chapter 5: Photochromic Molecular Registers*. As the eNBIPS sample is homogenous, the measurements made for initialising the registers are only carried out for one tape cell register, and reused for all other registers. The required steps include:

   a. Noise and capacity determination for tape cell register, determination of value mid points, and value ranges.
   b. Determination of the speed of colouration/decolouration and the creation of the value transition table.
   c. Resetting all tape cell registers to $V_0$ (representing blank).
2. *Observe*. Operates on: Head, Tape. This requires reading the symbol written to the cell the head is currently addressing. This is equivalent to reading a register as described in *Chapter 5: Photochromic Molecular Registers*. Though the read pulse uses decrement reading and hence decrements the tape cell register by one value, the register is always subsequently rewritten during the *print* operation.
3. *Decide*. Operates on: I-State, Transition Function. Cross referencing the I-State and value of the current cell with the transition function to obtain the instructions to enact. If the I-state is on the tape, the photochromic Turing machine head must move to and read the I-state register. Though reading the I-state changes the value, the I-state will be rewritten during the *update* operation. Taking the cell value from the observe action and the I-state value produces a three-tuple of actions from the transition function.
4. *Print*. Operates on: Head, Tape. Updating the value stored by a cell. In this implementation this is equivalent to writing to a register. The current value of the tape cell register is known (the value recorded during the observe action, minus one) and the desired value is given by the transition function. The value transition table is then referenced for the appropriate amount of colouration or decolouration stimulus to write the register to the new value.
5. *Move*. Operates on: Head. Moving the head relative to the tape by one cell left or right. This is implemented by incrementing or decrementing the position of the stepper motor by the cell size. The move action is also extended to allow for the head to move to the left-most cell to read the I-state register. Attempts to move beyond the bounds of the tape will result in the termination of the Turing machine with an '*exceeded tape bound*' error.
6. *Update*. Operates on: I-State. Updates the I-state to a new value determined by the transition function. If the I-state is stored on the controlling computer, this is altering a LabVIEW variable. If the I-state is instead stored on the

tape, the head moves to the left-most cell and alters the register value to the new I-state.

7. *Output*. Operates on: Tape, Head. The output of a Turing machine is the value of each register on the tape. The Output action reads every register on the cell, and presents it visually to the user or in a text log.

The flowchart of these actions can be seen in Figure 6.3.

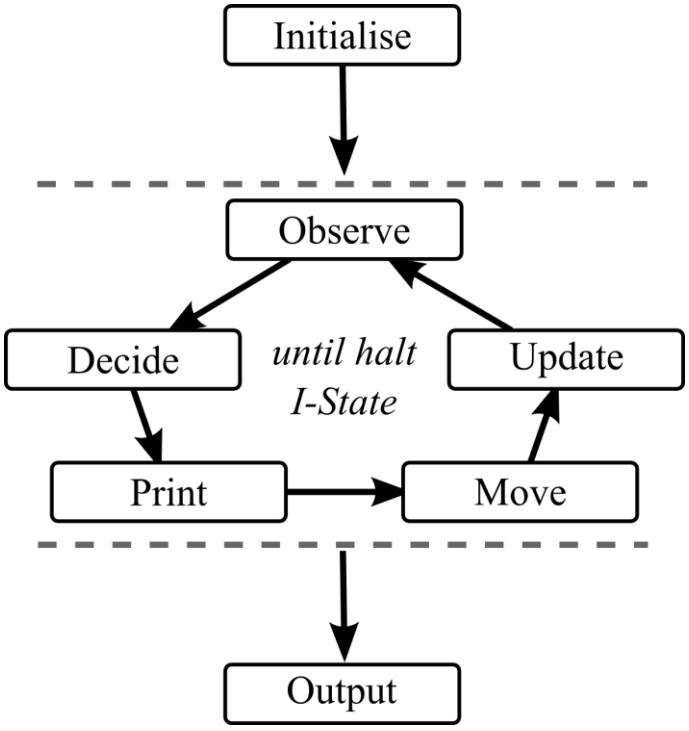



*Figure 6.3: Flowchart of photochromic tape Turing machine execution. The five fundamental actions transition between one another by the arrows. The compound actions* Initialise *and* Output *are separated from the main body of execution.*

## 6.2. Programming Busy Beavers in Photochromic Tape Turing Machines

Busy beavers [Radò (1962)] are examples of Turing machines where the machine performs the maximum amount of ‘work’ before halting when run on an unbounded tape with all cells set to blank. There are two definitions of work; the most written non-blank symbols, and the most move actions. These are given by the Generalised Busy Beaver function, and the Generalised Maximum Shifts function. Both functions are non-computable as they are equivalent to the halting problem; as they represent the maximum work by a machine *that still halts*, any machine that writes more non-blank symbols or makes more shifts than the champion busy beaver

will never halt [Radò (1962)]. The only way to prove a busy beaver is the champion is to enumerate every possible Turing machine for that class.

Busy beavers were chosen as the Turing machines to be executed by our experimental setup as they represent the smallest machines that produce a maximally long output with a complex series of head movements and I-state transitions. They represent the ideal test that our implementation of the tape works, as they represent the hardest test for each class of Turing machine.

A Turing machine is often classified by the size of the alphabet and the number of possible I-states. This is typically rendered as an '*n*-state *m*-symbol machine' where *n* and *m* are integers corresponding to |Q|-1 (omitting the halting I-state) and |Γ| respectively.

The generalised busy beaver function for an *n*-state *m*-symbol machine $\Sigma(n, m)$ is the number of non-blank symbols on the tape when the machine halts. The generalised maximum shifts function $S(n, m)$ is the number of *move* actions performed by the machine before halting. For each class of *n*-state *m*-symbol Turing machine, there exists at least one 'champion' machine for each work function; the machine for which the values of *S* or $\Sigma$ is greatest. There may be several machines tied for the title of champion and the same machine may be champion for both $\Sigma$ and *S*.

### 6.3. Execution of Busy Beavers on Photochromic Tape Turing Machines

The simplest busy beaver Turing machine is the 1-state 2-symbol champion for Σ and S, shown in Figure 6.4. For this figure (and all future Turing machine figures) a) shows the transition function for that Turing machine, and b) shows the execution of the photochromic Turing machine for the given transition function. The left-most column shows the current I-state, and the centre grid column shows the tape and head (the bolded cell). As the Turing machines are two-symbol machines, the cells are either '0' (white) or '1' (grey). Note that the cell state is implicit; the machine may not have observed that cell this iteration. To the right of the state are four columns. Observe shows the waveform recorded when the current cell was observed. Write shows the state the cell should be updated to. The previous state and the new state are cross referenced with the state transition table, and the value passed to the appropriate LED. Move is the direction the head should move one cell in. The

final column is the new I-state. In this case the controlling computer stores this new value. The bottom section labelled ‘tape output’ is the cell values and read waveforms during the output compound action.

Experimentally, the busy beavers shown in Figure 6.4, Figure 6.5 and Figure 6.6 all utilise 0.67mM 1.15mm thick eNBIPS samples as described in *4.1: eNBIPS Samples*. Illumination times are normally automatically determined by the calibration function detailed in *5.5.1: Initialisation* for other examples of parallel registers. However, for implementing the tape in a Turing machine, the time a register must retain its value is not known in advance and varies, and hence a stored value of 1 may thermally relax over time back into 0. To alleviate this, the colouration times used are long (60 seconds), sufficient to colourise the register close to its maximum state. This increases the time necessary for the register to thermally relax to an incorrect value.

Figure 6.4 is the 1-state 2-symbol busy beaver champion for $\Sigma$ and $S$, with values $\Sigma$=1 and $S$=1. The halting I-state is ‘1’. There are many tied champions for the 1-state 2-symbol class. This example busy beaver prints ‘1’, moves right and halts. The tape output row shows this, with a single one ($\Sigma$=1) printed in one step ($S$=1).

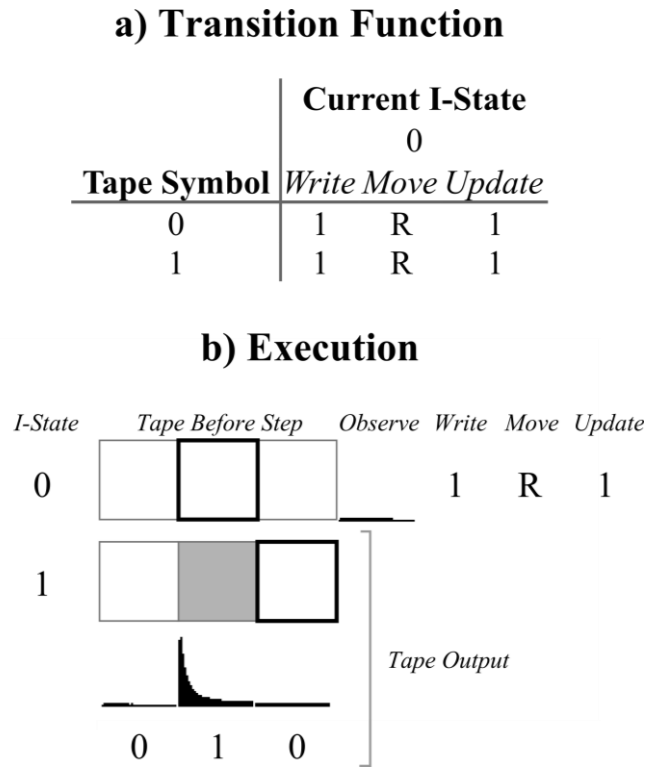



*Figure 6.4: The 1-state 2-symbol busy beaver champion for* Σ *and* S.

Figure 6.5 and Figure 6.6 show two more busy beavers for Turing machines with larger I-state sets. In both cases, the I-state is now stored in the tape itself, in a two-register cell to the left extreme of the tape. The Turing machines now read and write to this separated region every *decide* and *update* action.

Figure 6.5 is 2-state 2-symbol busy beaver champion for Σ and S, with values Σ=4 and S=6. The halting I-state is '2'. Different to the previous example, the Turing machine now stores the state on the tape as shown to the left, followed by the waveforms measured when the state is read using the Observe action. Note that the state register uses two digits as it has three I-states to store if you include the halting I-state, and the most significant bit is to the right. The tape output row shows the state of the tape at the end of execution, and is the correct output for this busy beaver, with 4 ones (Σ=4) after six steps (S=6).

**a) Transition Function**

| | Current I-State | | | | | |
|---|---|---|---|---|---|---|
| | 0 | | | 1 | | |
| **Tape Symbol** | *Write* | *Move* | *Update* | *Write* | *Move* | *Update* |
| 0 | 1 | R | 1 | 1 | L | 0 |
| 1 | 1 | L | 1 | 1 | R | 2 |

**b) Execution**

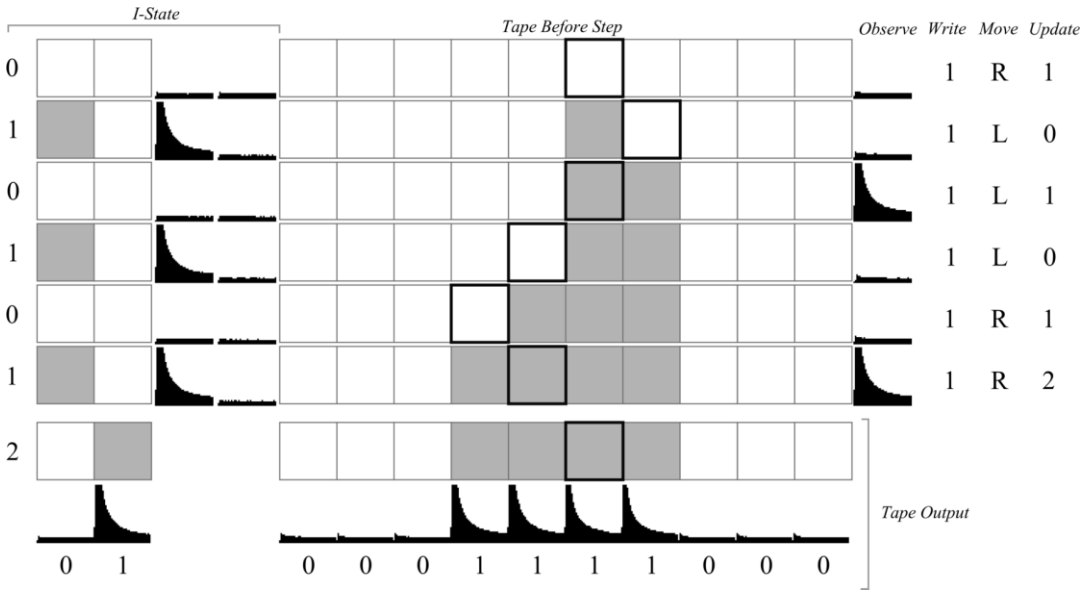



*Figure 6.5: The 2-state 2-symbol busy beaver champion for* Σ *and* S.

Figure 6.6 shows the 3-state 2-symbol busy beaver champion for Σ (but not S), with Σ=6 and S=14. The halting I-state is '3'. The tape output row shows the

state of the tape at the end of execution, and is the correct output for this busy beaver, with six ones (as Σ=6) after 14 steps (as S=14).

**a) Transition Function**

| | Current I-State | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | 0 | | | 1 | | | 2 | | |
| **Tape Symbol** | *Write* | *Move* | *Update* | *Write* | *Move* | *Update* | *Write* | *Move* | *Update* |
| 0 | 1 | R | 1 | 0 | R | 2 | 1 | L | 2 |
| 1 | 1 | R | 3 | 1 | R | 1 | 1 | L | 0 |

**b) Execution**

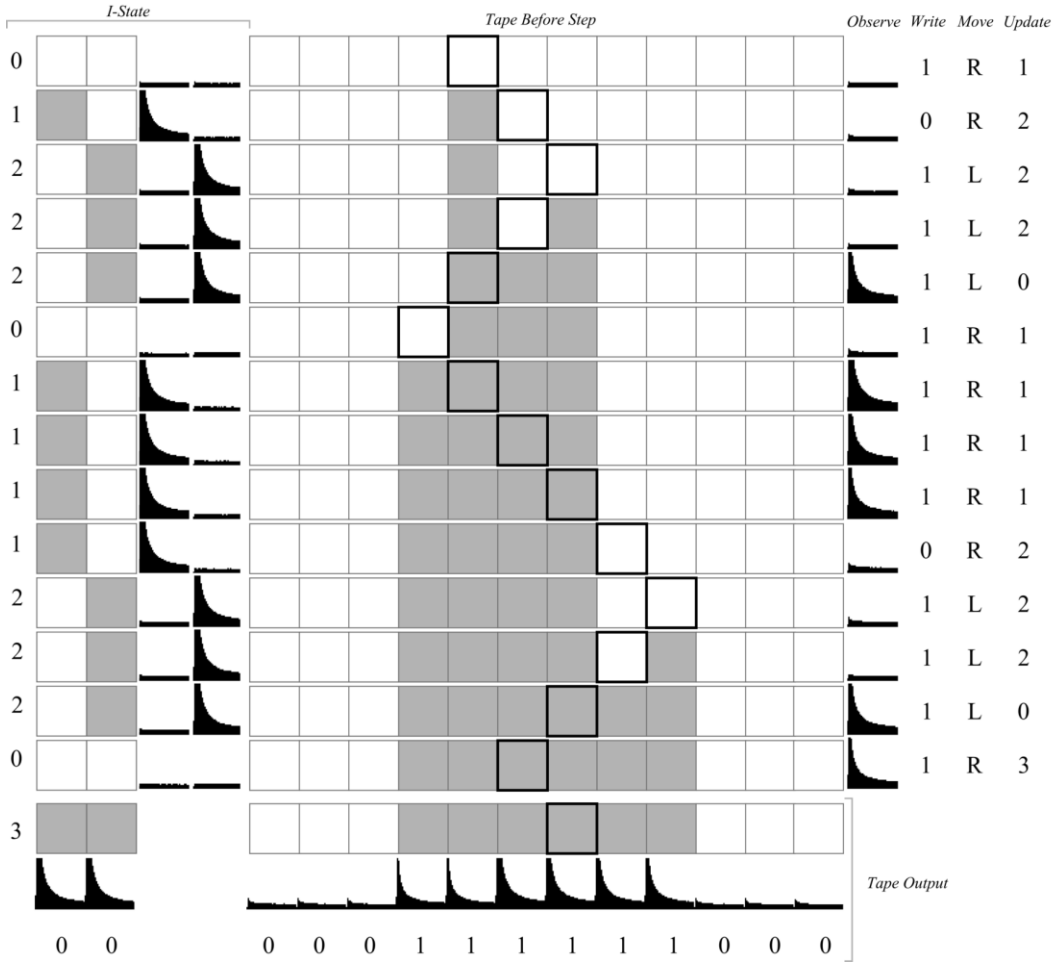



*Figure 6.6: The 3-state 2-symbol busy beaver champion for* Σ *(but not* S*).*

## 6.4. Discussion

In this chapter we have demonstrated how Turing machines can be implemented using arrays of eNBIPS registers forming the tape, with a movable illuminating head. Examples of busy beavers have been demonstrated in order to show how complex transition functions can be implemented. The serial nature of Turing machines lends itself well to the serial addressing of eNBIPS registers. The

fact that register values are altered by reading is not an issue, as cell values are always rewritten. The implementation of a Turing machine on a single homogeneous eNBIPS sample shows how sub-regions of spiropyran-doped substances can be individually addressed and store data to regulate function. The extension of parallel registers to Turing machines shows how photochromic registers can be applied to complex use patterns.

Photochromic-tape Turing machines could be extended to store the transition function on the tape. Like storing the I-state, this involves segregating a region on the left-side of the tape. Though this thesis has rendered the Turing transition functions as tables, they could equally be rendered as a set of tuples. Each tuple is of the form ($a \in \Gamma$, $b \in Q \rightarrow c \in \Gamma$, {R, L}, $d \in Q$), where *a* is the observed symbol, *b* is the current I-state, *c* is the printed symbol, R/L is the direction of movement, and *d* is the updated I-state. For example, the transition table for the busy beaver in Figure 6.4 could instead be written as two tuples: {(0, 0→1, R, 1), (0, 1→1, R, 1)}. This format lends itself well to tape transcription, as seen in Figure 6.7.

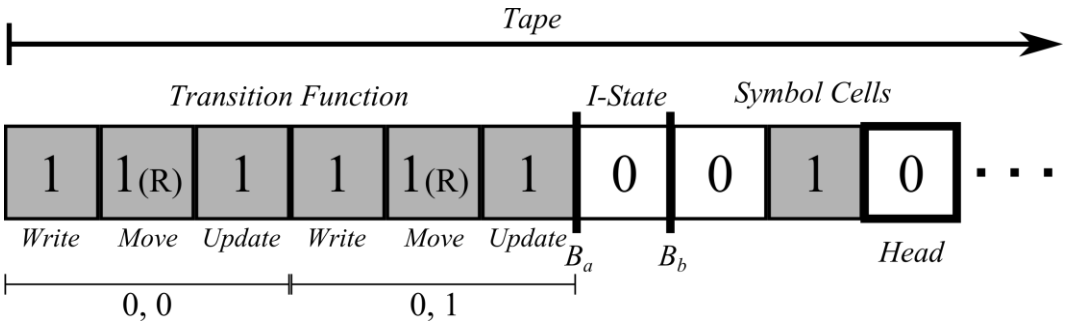



*Figure 6.7: One possible representation of a transition function stored on tape, with the transition function, I-state and Turing tape all on the same eNBIPS sample. Two virtual boundaries* $B_a$ *and* $B_b$ *subdivide the tape into three. The tuples are unlabeled in this example, their position is explicit; but tuples could also have labels at the cost of additional space. The L/R symbols are represented as 0 (left) and 1 (right).*

An issue with photochromic Turing machines is the length of the tape, limited by the number of parallel registers possible on an eNBIPS sample. We discuss methods to improve this point in *Chapter 9: Discussion and Conclusions,* but any method that improves tape capacity must also account for that I-state and possibly transition function being stored on that state. Our method places these at the

far-left of the tape, but alternative locations (such as offset in a second dimension) may be desirable depending on the nature of the improvements.

Additionally, the slow speed of *move* actions and duration of light pulses necessary to compensate for thermal relaxation leaves cells un-read for long durations. It is hence an unfortunate consequence that strengthening the system to thermal relaxation by increasing colouration durations also worsens the effect of thermal relaxation by increasing the duration a register can go unread. Additional experiments could be run to test the balance between longer and shorter colouration times, and their effect on register reliability. The experiments detailed in this section are reliable and repeatable as they have comparatively short execution times. Turing machines which take longer to execute, or with a higher number of symbols to store may occasionally have a cell that goes unread for a long period of time. Thermal relaxation of cells means these unread cells with higher values degrade to lower values, particularly if the cell is storing a higher base than binary. The typical symptom of this mode of failure is the Turing machine exceeding the bounds of the tape as it behaves unpredictably. Again, strategies to improve this are discussed in *Chapter 9: Discussion and Conclusions*.

# Chapter 7: Photochromic Molecular Logic Gates and Logic Circuits

The implementation of logic gates is an extension to registers, and could be applied to any photochromic molecule or other molecular switch with similar properties to NitroBIPS. In this chapter the details of extending the functionality of registers to cover a broad variety of logic gates are presented, including both two-input universal gates; NAND and NOR. This chapter then details the extension of gates into logic circuits.

## 7.1. Introduction

A finitary Boolean function is a mathematical function, taking a finite number *k* of Boolean inputs (the arity), and returning a single Boolean result. There are $2^{2^k}$ possible Boolean functions for each arity *k*. Claude Shannon demonstrated how two-value electrical switching circuits could implement Boolean functions

[Shannon (1938)], but Boolean logic is not restricted to electronic signals. Modern logic gates are typically constructed from silicon transistors, but implementations exist as diverse as carbon nanotubes [Bachtold et al. (2001)], quantum dots [Loss and DiVincenzo (1998)], rotaxane molecular architectures [Collier et al. (1999)], prokaryotic transcriptional networks [Silva-Rocha and de Lorenzo (2008)], smart polymers [Pasparakis et al. (2009)], bacteriorhodopsin films [Rao et al. (1996)] and DNA [Elbaz et al. (2010)].

The miniaturisation of logic components is a crucial part of modern technology. Billions of dollars are spent on chip fabrication techniques, resolving feature sizes as low as 14nm or smaller [Singh (2011)]. However, limitations on the possible size of components [Keyes (2001)] [Zhirnov et al. (2003)] mean this approach may be reaching its limits.

The examples of unconventionally implemented logic gates above are all very small or have the potential to be made very small. In some cases the examples may be looking for ways to increase processor speed or component density. In our case however, we are not seeking to compete with silicon processors in terms of computational power or speed. Instead, the use of molecules is with an aim to embed computation into biological systems, both natural and synthetic.

Logical processing at the molecular level takes place either at a uni-molecular scale, supramolecular scale or within populations of molecules. Photochromic molecules have already shown promise in implementing logic gates, including spiropyrans [Raymo and Giordani (2001)] and multiple covalently bonded species [Andréasson et al. (2011)], the latter implementing a remarkable 13 logical functions with a single molecule, giving rise to the possibility of light operated molecular species capable of implementing multiple functions.

To perform more complex logical functions, it is convenient to compose them of many logic gates connected to one another. This allows for the reuse of common components to build more complex entities. The connection between gates and the type of gates used forms the behaviour of the overall circuit.

This chapter shows how NBIPS can implement logical functions in the form of logic gates, how logic gates can implement logic circuits, and discusses the

limitations of optical inputs and outputs, why eNBIPS gates cannot communicate directly and methods by which this could be addressed.

## 7.2. Logic Gate Definitions

Our representation of logic gates is an extension of registers, and uses the representation detailed in 3.1.2. This section details the definitions of many types of logic gates, the sequences of pulses needed, and any additional properties.

### *7.2.1 1-Input Gates*

There are 4 ($2^{2^1}$) possible 1-input logic gates. Commutability is not applicable for 1-input gates as they only have a single input. All 1-input logic gates have equality of output.

**1-FALSE**

*A.K.A: Pass 0, NO.*

**Representation:** [1+, *( )*, $\downarrow_0$]

Unique?: *Yes*

| *Input* | *Output* |
|---|---|
| *T* | *F* |
| *F* | *F* |

**1-TRUE**

*A.K.A: Pass 1, YES.*

**Representation:** [2+, *({*$\uparrow_1$*})*, $\downarrow_1$]

Unique?: *Yes*

| *Input* | *Output* |
|---|---|
| *T* | *T* |
| *F* | *T* |

**1-ID**

*A.K.A: IDENTITY, Pass, Wire.*

**Representation:** [2+, *({a,* $\uparrow_1$*})*, $\downarrow_1$]

Unique?: *Yes*

| *Input* | *Output* |
|---|---|
| *T* | *T* |
| *F* | *F* |

**1-NOT**

*A.K.A: !a, ¬a,* $\bar{a}$ *, Inverter, Negation.*

**Representation:** [2+, *({*$\uparrow_1$*}, {a,* $\downarrow_1$*})*, $\downarrow_1$]

Unique?: *Yes*

| *Input* | *Output* |
|---|---|
| *T* | *F* |
| *F* | *T* |

### *7.2.2 2-Input Gates*

The most commonly considered gate type is the 2-input gates of binary logical operators. There are 16 ($2^{2^2}$) possible 2-input gates. Two gates of note include NOR and NAND; all other logic gates (and hence all logic circuits) can be

simulated with a circuit of just NOR or just NAND gates, making them functionally complete.

Truth tables presented here show input *a* along the x axis, and *b* along the y axis. Gates that do not have corresponding input pulses for both inputs are not commutable and hence not applicable for parallelisation. Non-unique gates will have their output correspondence listed. The input phase of a commutable gate is underlined.

**2-FALSE**

*A.K.A: Pass 0, NO, Contradiction.*

**Representation:** [1+, *( )*, $\downarrow_0$]

Unique?: *Yes*

Equality of Output?: *Yes*

Commutable?: *N/A*

| | **T** | **F** |
|---|---|---|
| **T** | *F* | *F* |
| **F** | *F* | *F* |

**2-NOR**

*A.K.A: $a\downarrow b$, $a \triangledown b$, Pierce Arrow, Quine Dagger.*

**Representation:** [2+, *({$\uparrow_1$}, {$\underline{a, \downarrow_1}$}, {$\underline{b, \downarrow_1}$})*, $\downarrow_1$]

Unique?: *Yes*

Equality of Output?: *Yes*

Commutable?: *Yes*

| | **T** | **F** |
|---|---|---|
| **T** | *F* | *F* |
| **F** | *F* | *T* |

**2-B NOT A**

*A.K.A: Converse Nonimplication, $a \not\subset b$, $a \overleftarrow{\leftarrow} b$.*

**Representation:** [3+, *({$\uparrow_1$}, {a, $\downarrow_1$}, {b, $\uparrow_1$}, {$\downarrow_1$})*, $\downarrow_1$]

Unique?: *Yes*

Equality of Output?: *Yes*

Commutable?: $\underline{No}$

| | **T** | **F** |
|---|---|---|
| **T** | *F* | *T* |
| **F** | *F* | *F* |

**2-NOT A**

*A.K.A: !a, ¬a, Negation.*

**Representation:** [2+, *({$\uparrow_1$}, {a, $\downarrow_1$})*, $\downarrow_1$]

Unique?: *Yes*

Equality of Output?: *Yes*

Commutable?: *N/A*

| | **T** | **F** |
|---|---|---|
| **T** | *F* | *T* |
| **F** | *F* | *T* |

**2-ABJUNCTION**

*A.K.A: Material nonimplication, $a \not\rightarrow b$.*

**Representation:** [2+, *({a, $\uparrow_1$}, {b, $\downarrow_1$}), $\downarrow_1$*]

Unique?: *Yes*

Equality of Output?: *Yes*

Commutable?: *No*

| | **T** | **F** |
|---|---|---|
| **T** | *F* | *F* |
| **F** | *T* | *F* |

**2-NOT B**

*A.K.A: b!, ¬b, Negation.*

**Representation:** [2+, *({$\uparrow_1$}, {b, $\downarrow_1$}), $\downarrow_1$*]

Unique?: *Yes*

Equality of Output?: *Yes*

Commutable?: *N/A*

| | **T** | **F** |
|---|---|---|
| **T** | *F* | *F* |
| **F** | *T* | *T* |

**2-XOR**

*A.K.A: Exclusive Disjunction, $\oplus$, $\veebar$.*

**Representation:** [3+, *({a, $\uparrow_1$}, {b, $\uparrow_1$}), $\downarrow_2$*]

Unique?: *No. $V_1$ = True. $!V_1$ = False.*

Equality of Output?: *No, non-unique.*

Commutable?: *Yes*

| | **T** | **F** |
|---|---|---|
| **T** | *F* | *T* |
| **F** | *T* | *F* |

**2-NAND**

*A.K.A: a↑b, Sheffer Stroke, alternative denial.*

**Representation:** [3+, *{$\uparrow_2$}, {a, $\downarrow_1$}, {b, $\downarrow_1$}), $\downarrow_2$*]

Unique?: *Yes*

Equality of Output?: *No*

Commutable?: *Yes*

| | **T** | **F** |
|---|---|---|
| **T** | *F* | *T* |
| **F** | *T* | *T* |

**2-AND**

*A.K.A: a ∧b, a & b, logical conjunction.*

**Representation:** [*3+, ({a, $\uparrow_1$}, {b, $\uparrow_1$}, {$\downarrow_1$}), $\downarrow_1$*]

Unique?: *Yes*

Equality of Output?: *Yes*

Commutable?: *Yes*

| | **T** | **F** |
|---|---|---|
| **T** | *T* | *F* |
| **F** | *F* | *F* |

**2-XNOR**

| | **T** | **F** |
|---|---|---|
| **T** | *T* | *F* |
| **F** | *F* | *T* |

*A.K.A: a ↔ b, a = b, logical biconditional.*

**Representation:** [*3+, ({↑$_1$}, {a, ↑$_1$}, {b, ↓$_1$}), ↓$_2$*]

Unique?: *No. $V_1$ = True. !$V_1$ = False.*

Equality of Output?: *No, non-unique.*

Commutable?: *No.*

**2-ID A**

| | **T** | **F** |
|---|---|---|
| **T** | *T* | *T* |
| **F** | *F* | *F* |

*A.K.A: Pass a, projection function.*

**Representation:** [2+, *({a, ↑$_1$}), ↓$_1$*]

Unique?: *Yes*

Equality of Output?: *Yes*

Commutable?: *N/A*

**2-A IMP. B**

| | **T** | **F** |
|---|---|---|
| **T** | *T* | *T* |
| **F** | *F* | *T* |

*A.K.A: if a then b, material implication, a ⊃ b, a → b.*

**Representation:** [*3+, ({↑$_1$}, {a, ↑$_1$}, {b, ↓$_1$}), ↓$_2$*]

Unique?: *Yes*

Equality of Output?: *No*

Commutable?: *No*

**2-ID B**

| | **T** | **F** |
|---|---|---|
| **T** | *T* | *F* |
| **F** | *T* | *F* |

*A.K.A: Pass b, projection function.*

**Representation:** [2+, *({b, ↑$_1$}), ↓$_1$*]

Unique?: *Yes*

Equality of Output?: *Yes*

Commutable?: *N/A*

**2-B IMP. A**

| | **T** | **F** |
|---|---|---|
| **T** | *T* | *F* |
| **F** | *T* | *T* |

*A.K.A: if b then a, converse implication, a ⊂ b, a ← b.*

**Representation:** [*3+, ({↑$_1$}, {a, ↓$_1$}, {b, ↑$_1$}), ↓$_2$*]

Unique?: *Yes*

Equality of Output?: *No*

Commutable?: *No*

**2-OR**

| | **T** | **F** |
|---|---|---|
| **T** | *T* | *T* |
| **F** | *T* | *F* |

*A.K.A: logical disjunction, $a \vee b$, $a||b$.*

**Representation:** [3+, *({a, $\uparrow_1$}, {b, $\uparrow_1$}), $\downarrow_2$*]

**Alternative:** [2*!*, *({a, $\uparrow_1$}, {b, $\uparrow_1$}), $\downarrow_1$*]

*The alternative representation restricts the possible output states to just $V_0$ and $V_1$.*

Unique?: *Yes*

Equality of Output?: *No (Original Representation). Yes (Alternative Representation)*

Commutable?: *Yes*

**2-TRUE**

| | **T** | **F** |
|---|---|---|
| **T** | *T* | *T* |
| **F** | *T* | *T* |

*A.K.A: Pass 1, YES, Tautology.*

**Representation:** [2+, *({$\uparrow_1$}), $\downarrow_1$*]

Unique?: *Yes*

Equality of Output?: *Yes*

Commutable?: *N/A*

### *7.2.3 n-Input Gates*

Some gates which are commutable and unique can also be adapted to *n*-input gates, where *n* is any positive integer. An *n*-input gate can either have a pre-defined number of inputs, or in some cases an arbitrary number of inputs. Gates which can accommodate an arbitrary number of inputs can receive any number of inputs during the input phase without foreknowledge or alteration of the gate. Gates that cannot accommodate an arbitrary number of inputs require changes to the gate capacity, moderators or output pulse if the number of inputs changes. In the following definitions, *n* is the number of inputs and Ω is the *n*th unique identifier for inputs.

**n-FALSE**

*Truth Condition: False*

**Representation:** [1+, *( ), $\downarrow_0$*]

Arbitrary Number of Inputs?: *N/A*

Equality of Output?: *Yes*

**n-NOR**

*Truth Condition: If all inputs are False, then True. Else False.*

**Representation:** [2+, *({$\uparrow_1$}, {a, $\downarrow_1$}, {b, $\downarrow_1$}, ... {Ω, $\downarrow_1$}), $\downarrow_1$*]

Arbitrary Number of Inputs?: *Yes*

Equality of Output?: *Yes*

**n-NAND**

*Truth Condition: False only if all inputs are True. Else True.*

**Representation:** [(n+1)+, *({↑$_n$}, {a, ↓$_1$}, {b, ↓$_1$}, ... {Ω, ↓$_1$}),* ↓$_n$]

Arbitrary Number of Inputs?: *No*

Equality of Output?: *No*

**n-AND**

*Truth Condition: True only if all inputs are True. Else False.*

**Representation:** [(n+1)+, *({a, ↑$_1$}, {b, ↑$_1$}, ... {Ω, ↑$_1$}, {↓$_{n-1}$}),* ↓$_1$]

Arbitrary Number of Inputs?: *No*

Equality of Output?: *Yes*

**n-OR**

*Truth Condition: True if one or more inputs are True. Else False.*

**Representation:** [(n+1)+, *({a, ↑$_1$}, {b, ↑$_1$}, ... {Ω, ↑$_1$}),* ↓$_n$]

Arbitrary Number of Inputs?: *No*

Equality of Output?: *No*

**Alternative:** [2*!*, *({a, ↑$_1$}, {b, ↑$_1$}, ... {Ω, ↑$_1$}),* ↓$_1$]

Arbitrary Number of Inputs?: *Yes*

Equality of Output?: *Yes*

**n-TRUE**

*Truth Condition: True*

**Representation:** [2+, *({↑$_1$}),* ↓$_1$]

Arbitrary Number of Inputs?: N/A

Equality of Output?: *Yes*

## 7.3. Implementation of Photochromic Logic Gates

### *7.3.1 Experimental Setup*

Logic gates use the same eNBIPS samples and hardware set-up as registers. The difference is the sequence of light pulses to which they are exposed. To control the sequence of pulses, LabVIEW code was written. In anticipation of future expansion to logic circuits and elementary cellular automata, logic gates were implemented as modular subVIs that could be easily inserted into existing code.

A LabVIEW Logic Gate takes as input:

- The DAQmx tasks and associated error clusters: These allow the sub-VI to operate the LEDs and measure from the photodiode.
- The Boolean inputs to the gate: The value of these determine if the input pulses occur or not.
- The measurement sampling rate. This is how frequent the photodiode voltage is sampled.
- Average background noise. This allows us to cut short decolourations that have already fully decoloured the sample by checking if the fluorescence has reached background levels, increasing the speed of execution.
- Truth threshold: The fluorescence value above which the output is considered *True*. We only implement unique gates, so a lower threshold is sufficient. This could be considered the lower bound of $V_1$ as described with registers.
- Averaging time: To protect against noise spikes, the peak fluorescence during the output phase is averaged over a given time, which can be set.
- Ultraviolet and Green LED Voltages.
- Timings. For most gates, this is two values. The duration of colouration stimulus to enact $\uparrow_1$, and the amount of decolouration stimulus to enact $\downarrow_1$. For same gates there may be additional timings, such as $\uparrow_2$ for 2-NAND (which is not equal to $2\times\uparrow_1$ as discussed in Photochromic Molecular Registers). Any program including logic gates will include a logic gate calibration subVI, which determines the rate of colouration and decolouration, and produces the shortest timings necessary for the gates to work reliably. Gates are then tested using the timings to ensure they produce the expected results. If the gates fail, the timings are increased and tested again, until a reliable yet fast execution time is established.

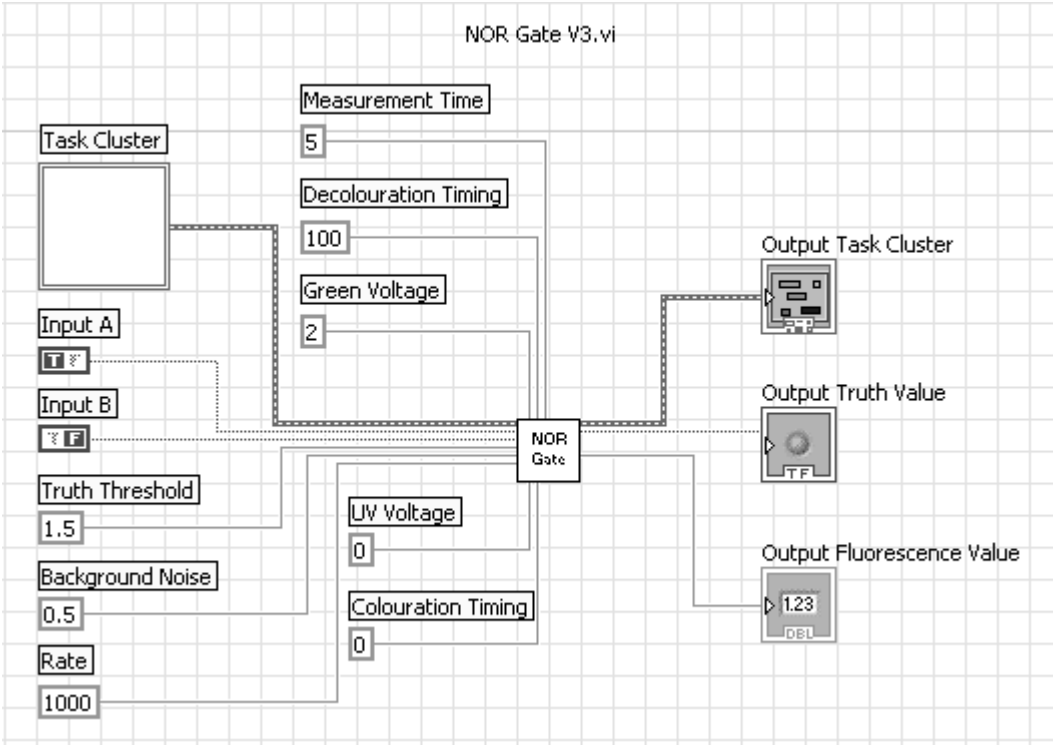



*Figure 7.1: A Logic Gate subVI in LabVIEW.*

The outputs from the Gate subVI are the output truth value, the completed DAQmx tasks to be used by the next gate, and the recorded measurements from the photodiode (for diagnostic purposes), as shown in Figure 7.1.

The subVI itself is a state machine. A LabVIEW state machine is a design pattern consisting of multiple ‘cases’ implementing functionality, and controlled transitions between them, summarised in Figure 7.2. The Gate subVI consists of an initialisation case to start the optical tasks; a final output case that determines the output value by comparing the peak fluorescence during the output pulse with the truth threshold; and a case for every light pulse in the gate’s definition.

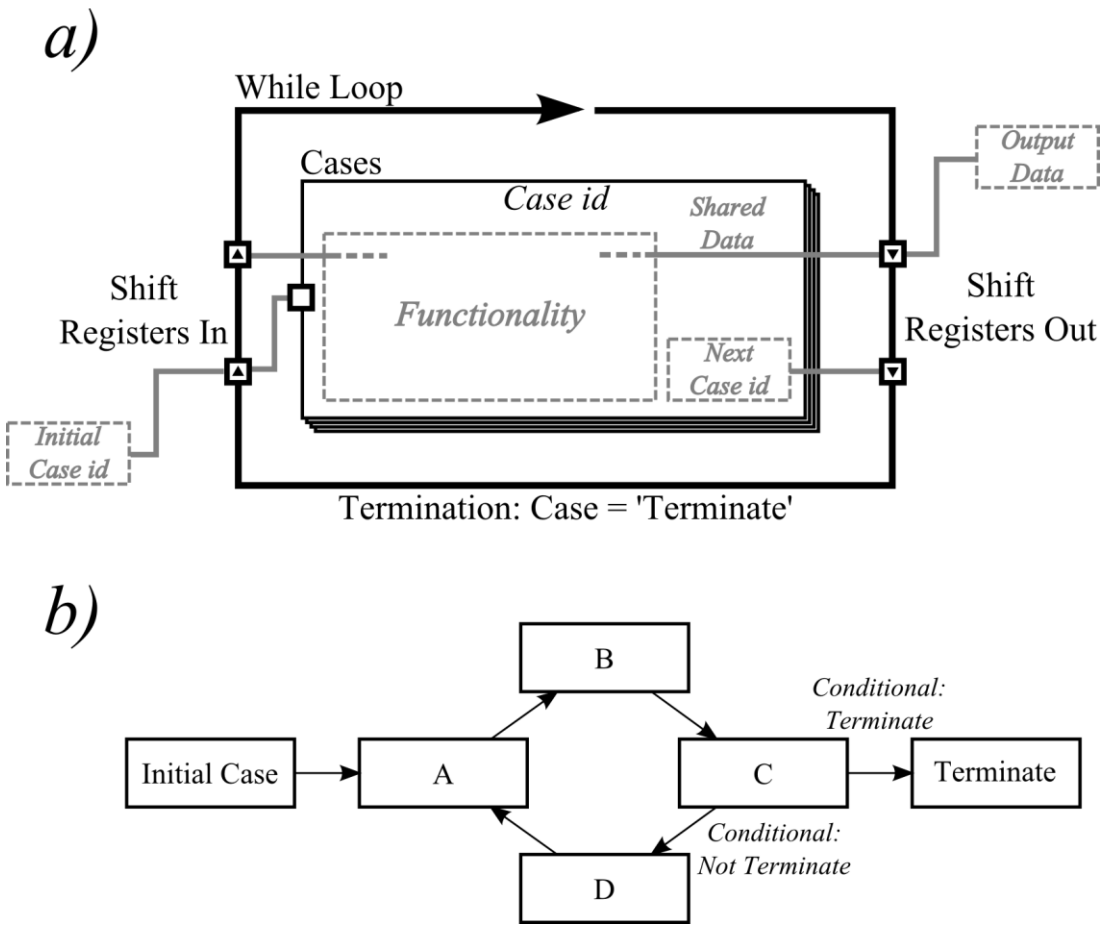



*Figure 7.2: Overview of LabVIEW state machines. a) Program pattern of a LabVIEW state machine. b) State diagram showing how cases transition between one another, possibly with conditional dependencies.*

Input pulse cases contain a conditional that only operates the pulse if the corresponding input is true. The pulse is enacted by the colouration / decolouration subVIs described in *Chapter 4: Methods*. Moderator pulse cases do not have a conditional, just the colouration / decolouration sub VI. The output pulse case contains the modified decolouration subVI which averages the recorded fluorescence to be more resilient to noise. Decolouration pulses may use modified decolouration subVIs which cease the pulse early if the recorded fluorescence is equal to the level of background noise. This decreases execution time by not attempting to decolour gates which were already in $V_0$. The case structure of a 2-input gate subVI is shown in Figure 7.3.

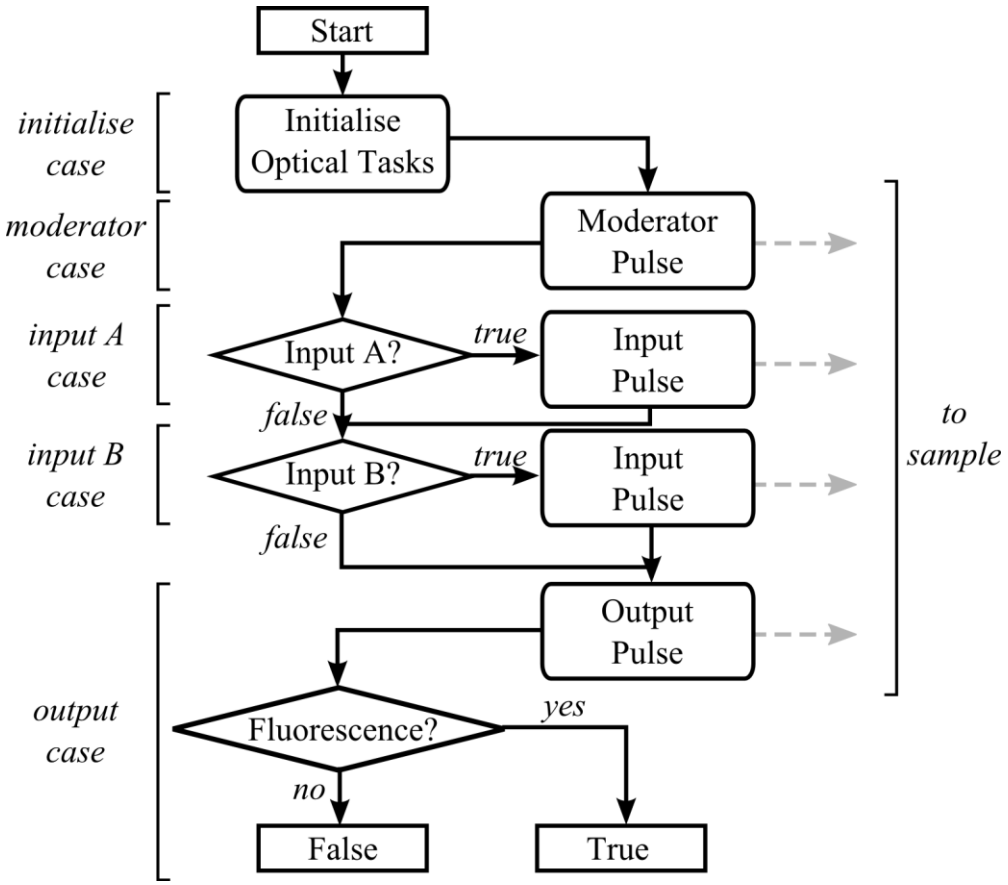



*Figure 7.3: Flowchart of logic gate subVI execution. This example is a 2-NOR gate.*

Gates with an arbitrary number of inputs do not have fixed Boolean inputs. Instead, an array of Boolean values representing the inputs is passed. The input phase is then internally implemented with a looped case which executes once for each value in the array, enacting an input pulse (or not) for each.

Like registers, noise limits the performance of a logic gate. However, whereas noise reduces the storage capacity of registers, noise reduces the execution speed of logic gates. As a logic gate (typically) only transitions between two or three values, the closer the value mid points are on the fluorescence range, the less light is needed for the transitions; fewer molecules need to switch form to change the sample value. As our LEDs operate at a fixed intensity, less light implies shorter pulse durations.

To help improve both the speed and reliability of our system, a logic gate pulse-time calibrator was implemented. This simple LabVIEW SubVI takes a logic-gate SubVI of choice, and using a user-set minimum pulse time executes all possible

input variations to that gate (and repeats this process a user-specified number of times, typically 4), and compares the recorded outputs to the expected outputs. If the outputs do not match the expectations, the pulse times are increased (again, by a user-set increment), and the process repeats, until a stable yet short pulse duration can be reached. The examples of gates and circuits in this chapter do not make use of this facility to produce figures with clear fluorescence peaks, however the implementations of logic circuits in *Chapter 8: Photochromic Molecular Elementary Cellular Automata* do.

## 7.4. Example Executions of eNBIPS Logic Gates

The following figures (Figure 7.4 to Figure 7.8) are traces of fluorescent emission recorded from the experimental executions of a selection of eNBIPS logic gates. Each gate has multiple traces associated with it, to show a selection of possible inputs to the gate. ‘False’ input pulses do not occur, and the gate skips immediately to the next pulse. For clarity, these figures pad data with zeros to make false pulses more obvious. Additionally for clarity, the duration of pulses has been extended to exaggerate the fluorescence peaks. The output pulses used early ceasing based on background noise; as a result the output pulses may be of different durations.

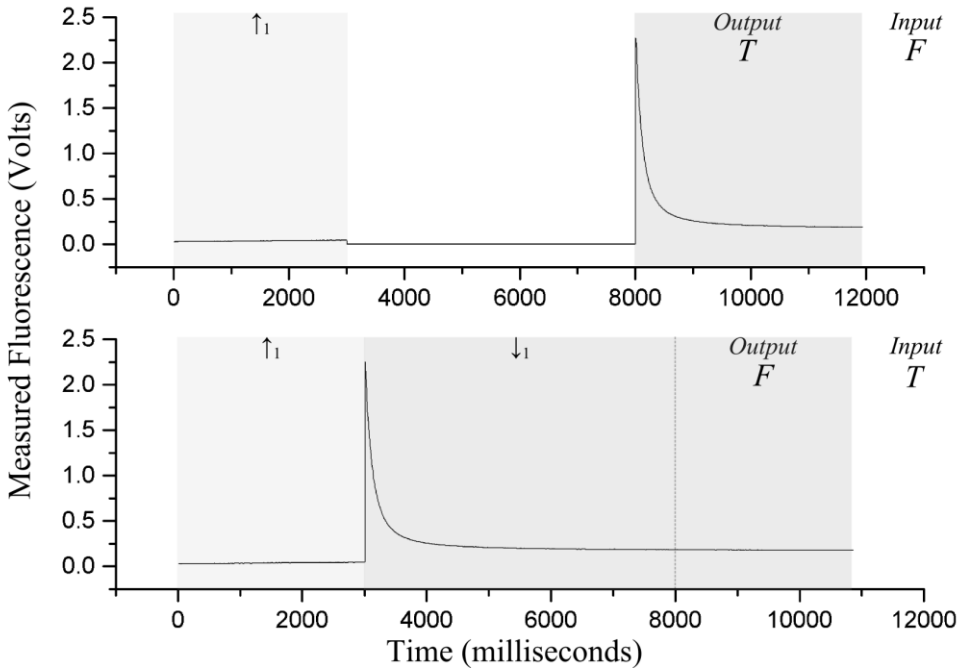



*Figure 7.4: Experimentally recorded traces from our system showing the two possible execution traces of a NOT gate.*

Figure 7.4 shows two executions of a NOT gate on our optical system. NOT is the most widely used of the 1-input gates, and represents the inversion of a signal. It was also the first gate implemented on our system. Only when the input is false, is the output true (fluorescence peak > 1 V). Pulse durations were 3000 ms for colouration, 5000 ms for decolouration. The traces show how the fluorescence peak always occurs during the gate execution, but the presence of the peak during the output pulse is the indicator of a true output.

Figure 7.5 shows four executions of a 2-NOR gate, executed on our system. Only when both inputs are false is the output true. Pulse durations used were the same as for NOT: 3000 ms for colouration and 5000 ms for decolouration. 2-NOR represents the two-input equivalent of a NOT gate. If either input is true, the output is false. If neither input is true, the output is true. By setting the gate to $V_1$, the gate will only remain in that state is neither input pulse occurs (being $\downarrow_1$ pulses). 2-NOR is used heavily during our experiments, as it is a universal gate, and only requires two values to operate (compared to three for NAND; the other universal gate).

Figure 7.6 shows four executions of a 2-OR gate in the standard representation with three values, executed on our system, with a colouration time of 3000ms. Either input pulse being true will colourise the sample to $V_1$, and both being true will colourise the sample to $V_2$, both representing true outputs. This is an example of a gate without equality of output; the fluorescence peaks of TF (or FT), and TT are different, despite both representing true. Unlike many gates, a 2-OR gate may have no fluorescence peak at all, in the FF case. Figure 7.7 shows a resolution to this; the alternative representation 2-OR gate. By extending the colouration time to 120,000ms, this gate essentially has only two values; $V_0$, and $V_{max}$. As a result, no colouration is caused by a second true input pulse, and the output fluorescence peaks are the same. However, this representation is inefficient due to the duration of pulses required, and as a result neither representation of 2-OR is frequently used in our experiments.

Figure 7.8 shows several executions of an n-NOR gate; an example of a gate with an arbitrary number of inputs. Pulse durations were 3000 ms for colouration and 5000 ms for decolouration. After the moderator pulse sets the n-NOR gate to $V_1$, any number of input pulses can occur before the output pulse. This allows for the n-NOR gate to be used as a 3-NOR gate (as shown with the first eight traces), or any long string of inputs, as shown in the last trace.

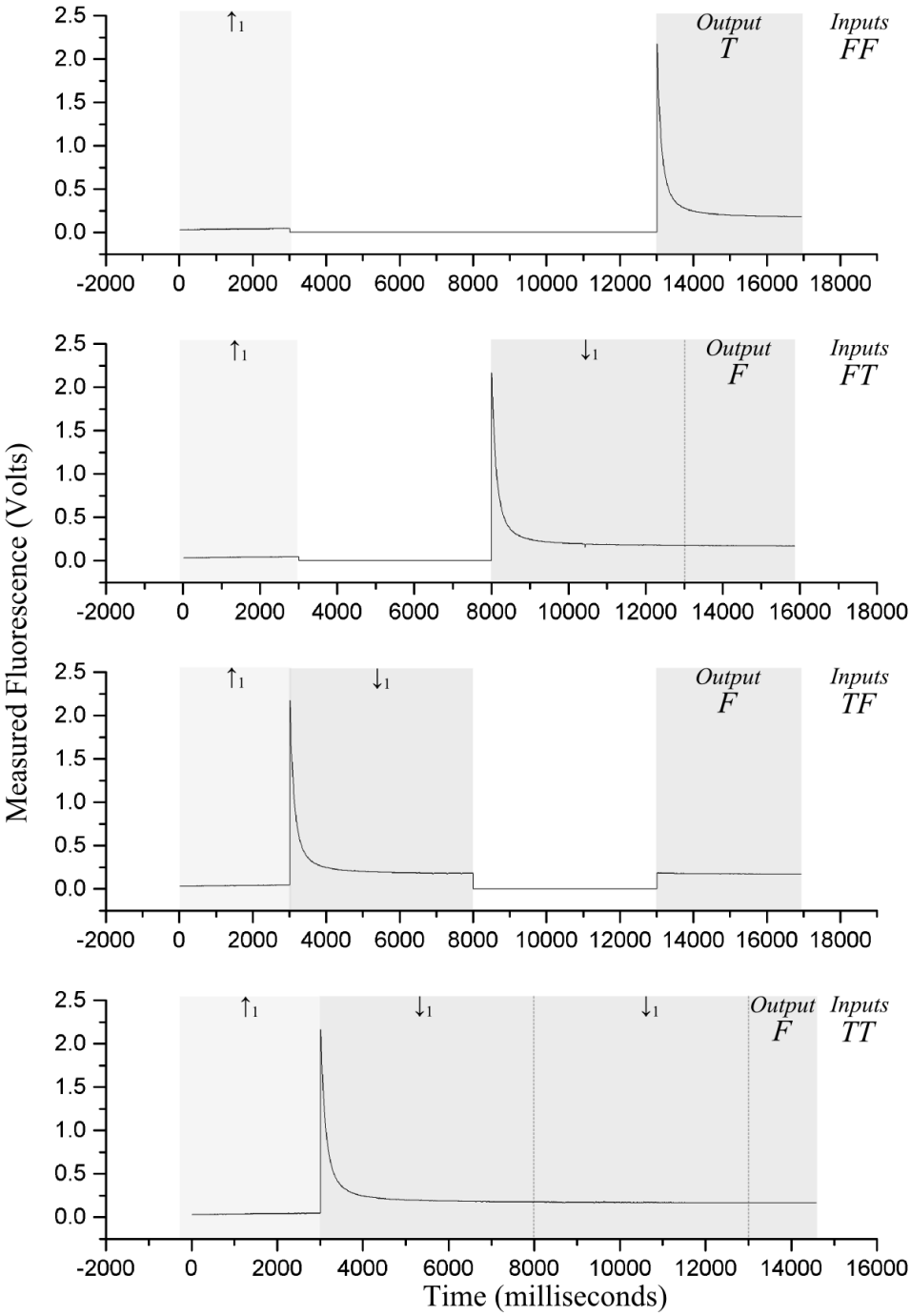



*Figure 7.5: Experimentally recorded traces from our system showing the four possible traces of a 2-NOR gate.*

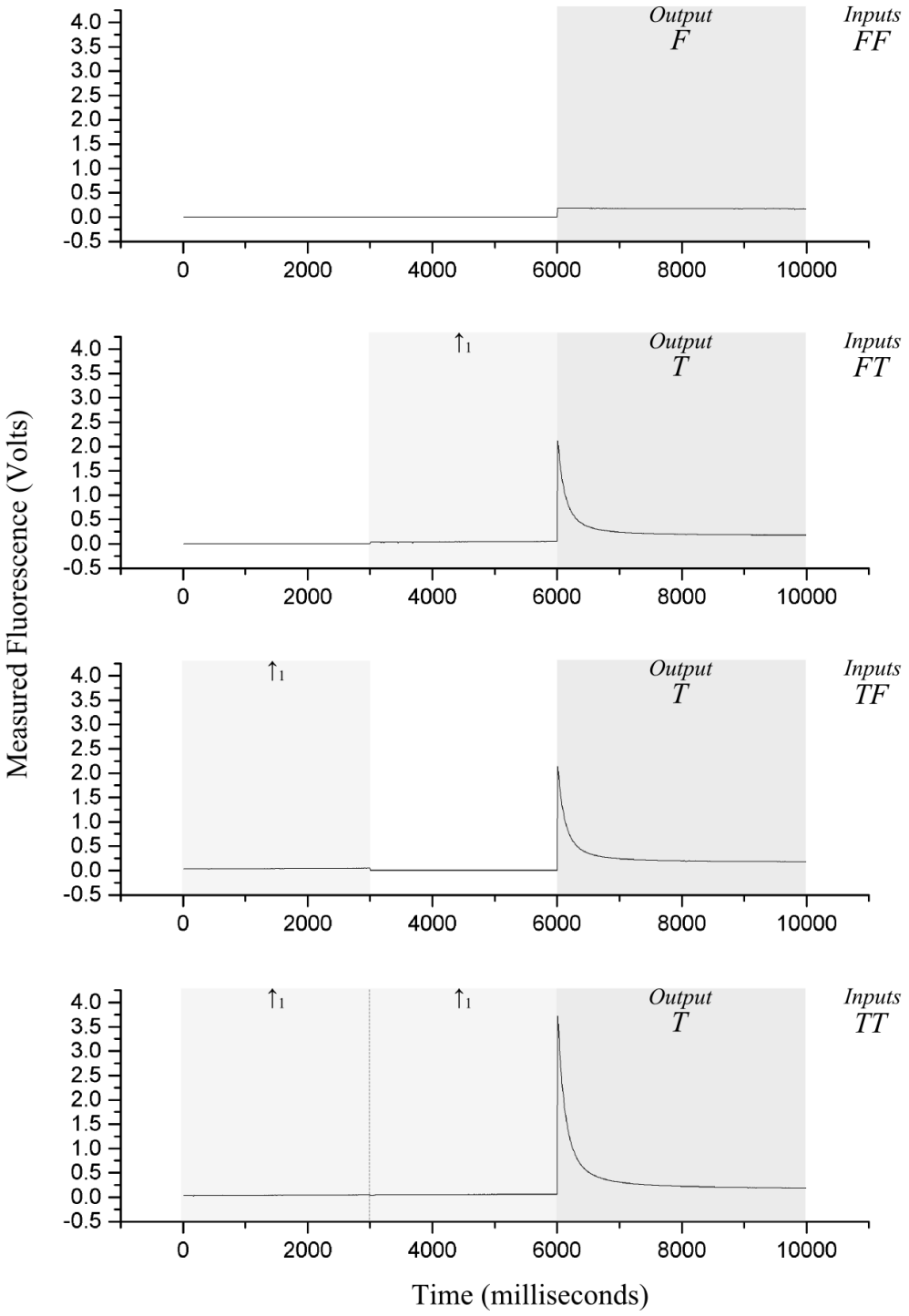



*Figure 7.6: Experimentally recorded traces from our system showing the four possible traces of a 2-OR gate.*

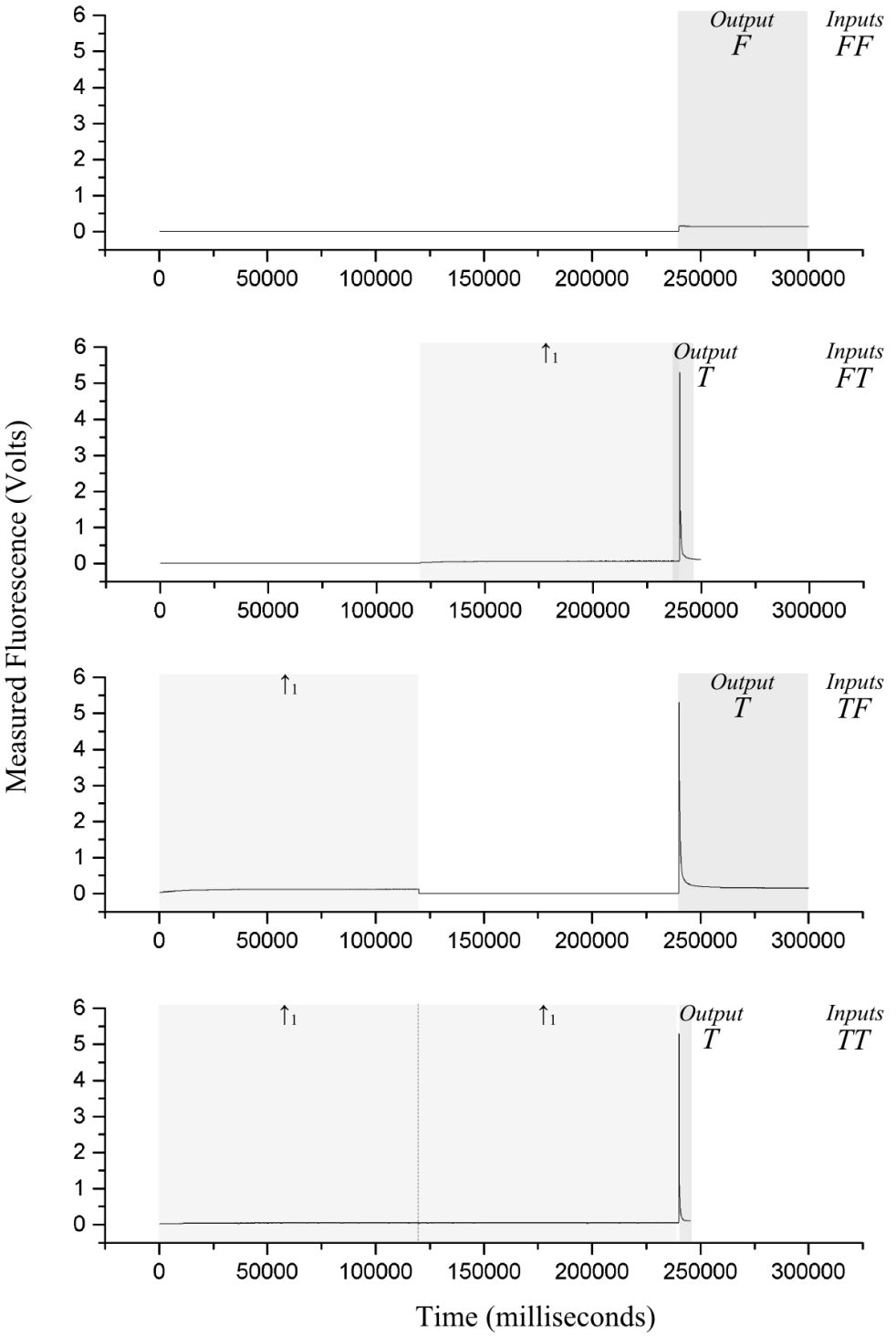



*Figure 7.7: Alternative representation of a 2-OR gate with only two states.*

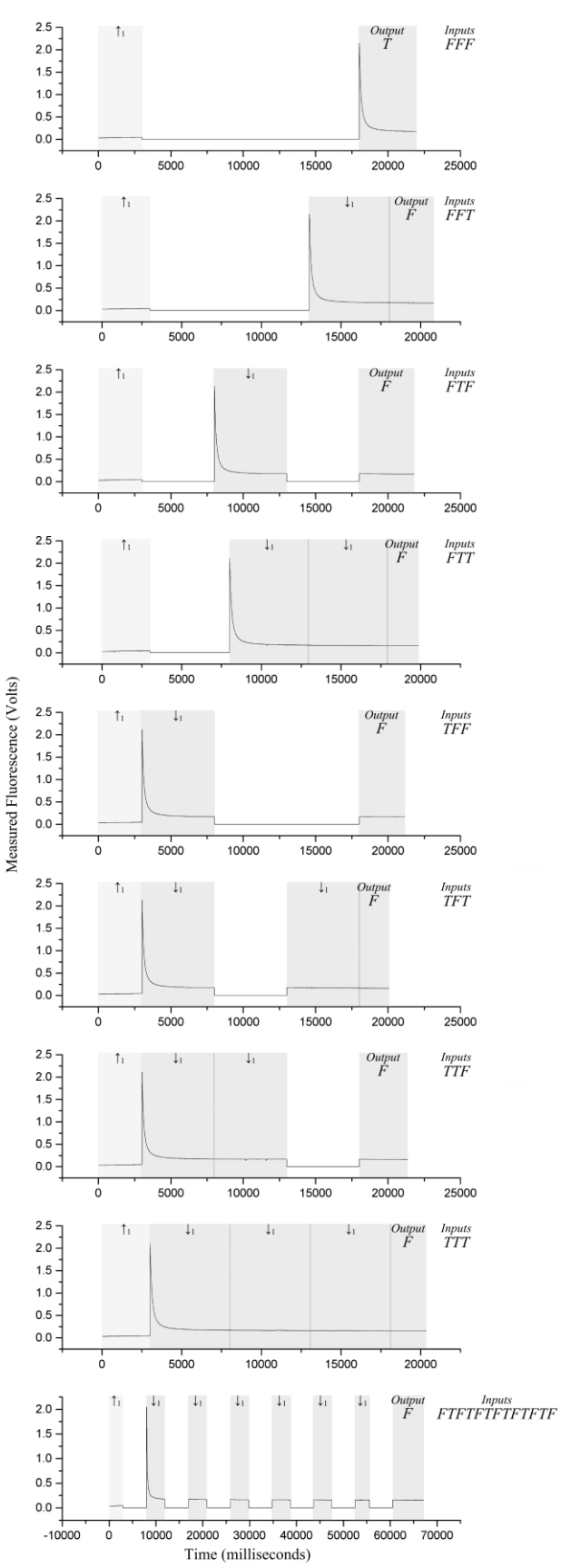



*Figure 7.8: Experimentally recorded traces from our system showing nine examples of an n-NOR gate.*

In comparison to the repeatability issues mentioned in *5.5.4 Parallel Registers*, the repeatability of logic gates is much more solid. The figures in this section use pulse durations in excess of those necessary, to give clear peaks. Shorter pulses can be used for gates using the floor restriction. (i.e. all except the alternative 2-OR gate). As a gate is not required to store data for a long, unpredictable amount of time and because the gate does not have to be colourised towards the maximum-MC state, very little thermal relaxation occurs.

## 7.5. Example Circuit Executions

Gates are extended to circuits using the techniques detailed in *3.2: Programming the System*. Example executions of circuits on our experimental setups can be seen in Figure 7.9 and Figure 7.10. Figure 7.9a shows executions of a drag-and-drop circuit $C_{ex}$ (Figure 3.3). A drag-and-drop circuit, if correctly implemented, shows no difference in its execution from a compiled circuit. Each trace is the level of fluorescence over time when exposed to light pulses (shaded regions). For clarity, pulses which are not enacted as their corresponding input was false instead pause for the pulse duration so the traces align. Additionally, a one second gap was added to show where pulses begin and end. T and F labels are added to the output pulses for clarity. Due to the exponential nature of NitroBIPS colouration and decolouration, the difference (in this case) between $V_0$ and $V_1$ is much smaller compared to the difference between $V_1$ and $V_2$. Figure 7.9b shows a zoomed view of a true output (grey) and a false output (black) to clarify this difference.

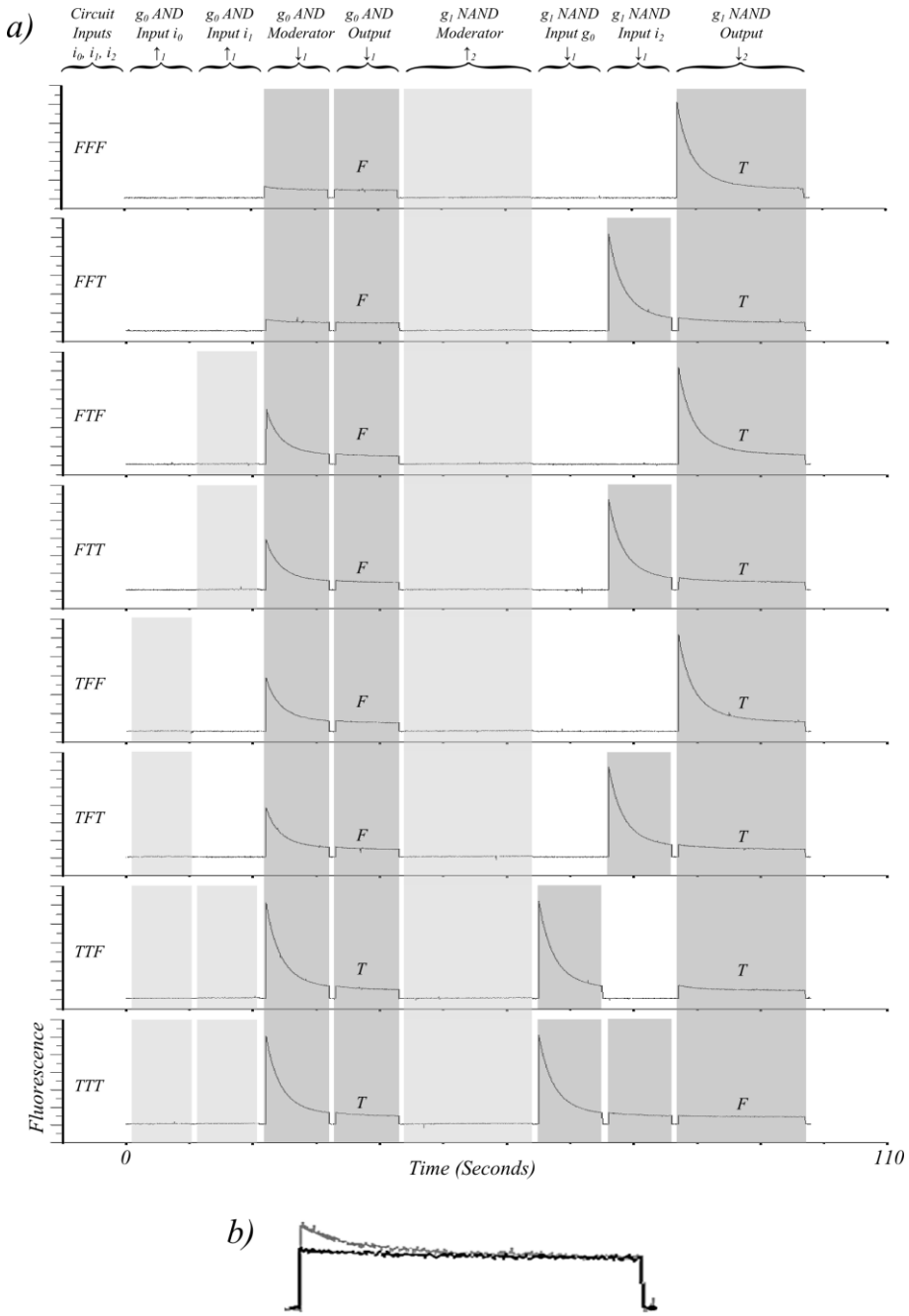



*Figure 7.9: Execution of a two-gate circuit* $C_{ex}$. *a) The fluorescence traces recorded experimentally from the eight possible executions of* $C_{ex}$. *b) This is a zoomed view of the difference between a True output (grey, from TTF) and a False output (black, from TTT).*

Figure 7.10 is an example of a compiled circuit, in this case executing a NOR-logic 2-bit full adder designed in Logisim (Figure 7.10a), and ran through our compilation pipeline described in *3.2.2: Compiled Circuits*. The gates were serialised by our compiler and converted into a pulse sequence for execution. It is calculating 3

+ 1, and the fluorescence trace is shown in Figure 7.10b[3]. Each gate is separated by a line, and is numbered and labelled with its truth value. $g_7$ and $g_{16}$ are false, and $g_{17}$ is true; indicating $o_c$ (the carry bit) is true, and hence the answer is 4.

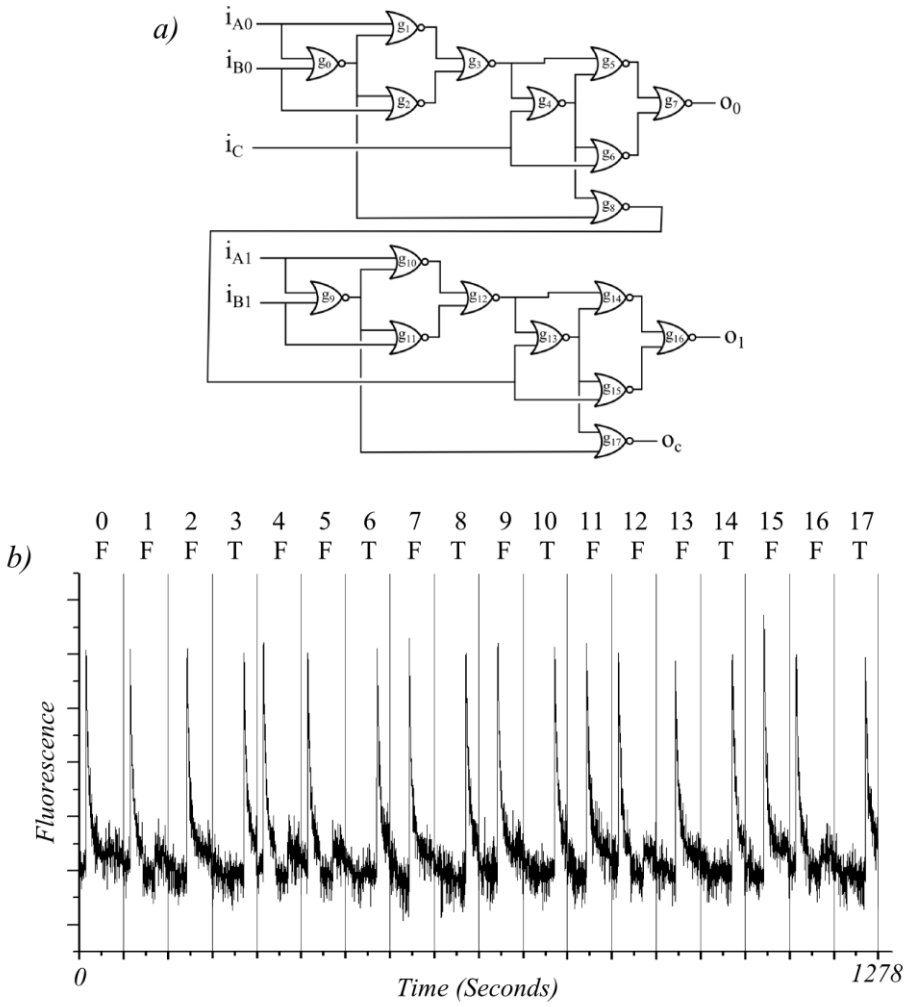



*Figure 7.10: Compilation and execution of a large logic circuit. a) A NOR-Logic 2-bit full adder designed in Logisim. b) The recorded output fluorescence trace for that circuit executing 3+1.*

[3] It should be noted that this figure's trace is considerably noisier than any other figures in this thesis. The data here is in fact the oldest in this thesis, and was recorded before the experimental setup moved laboratories. The move helped isolate the setup from environmental electromagnetic noise, primarily from air conditioning units in the ceiling (which later flooded the lab after freezing in winter).

## 7.6. Discussion

In this section we have demonstrated how eNBIPS samples can execute logic functions when exposed to stimulus, with light pulses as inputs and moderators, and fluorescence as output. These techniques could be applied to other photochromic molecules or in general to molecular switches.

The examples shown in 7.4 detail single logic gates. By illuminating different areas of an eNBIPS sample, multiple gates could be executed in parallel. As our experimental setup cannot illuminate multiple regions simultaneously this would be superfluous; the execution speed would not improve. However, by using a spatial light modulator as discussed in *Chapter 9: Discussion and Conclusions*, multiple gates could be executed in parallel with an execution time speedup.

Photochromic logic gates have a different mode of operation to traditional logic gates, relying on ordered sequences of input light pulses. For an experimental setup as described in this paper we can implement almost all logic functions by varying the order of pulses without modification to the eNBIPS sample itself. Similarly, the sample can be repurposed to act as a register or a logic gate by altering input light pulses. This offers the possibility of dynamic reallocation of resources between data storage and parallel execution of logic functions.

This section also demonstrates how photochromic logic circuits can be executed via the sequential execution of multiple photochromic logic gates. Though every gate is executed on the single eNBIPS sample, each gate can have a different function. The examples shown require serialisation, as our experimental setup cannot address more than one region of eNBIPS simultaneously. A setup which could illuminate multiple areas in parallel could execute multiple gates simultaneously, and remove the need for circuit serialisation.

The serial nature of our implementation means any sized circuit can be executed, in a time linear with the number of gates in that circuit. Each gate may have different execution times, depending on the number and type of light pulses required. Gates may also use early ceasing of the output pulse, which can alter gate duration. Gates are not held at a value for any length of time, so thermal relaxation is rarely an issue unless the pulse durations are set too low, especially with 3-value gates such as AND and NAND.

Gate failure will almost always manifest as a false output when a true output should have occurred, as the gate value has been decremented by thermal relaxation. More rarely, a large noise spike during the output pulse of a gate could incorrectly register the output as true rather than false. Assuming the threshold for a positive output has been correctly set; this has only been shown to occur due to mechanical trauma to the system (causing an environmental light bleed) or failure to set the filter to filter AC wiring noise.

The limit on circuit size is molecule bleaching. Over time, molecules will become bleached and the sample will lose dynamic range. Assuming the system periodically re-calibrates to account for this, the circuit can continue to be executed for a long time. How long is not currently known; circuits were looped for 48 hours over a weekend, and remained functional. If we use the gate durations from Figure 7.9 (i.e. 75 seconds per NOR gate), then the lower bound on maximum circuit size that could be achieved with our current setup is 2300 gates. The maximum is likely far higher, and could be improved with faster gate execution and sample replacement (i.e. moving the area of illumination if the sample is large).

A limitation of photochromic optical logic circuits is that each molecule's conversion of energy from input to fluorescence is not 100% efficient. Stoke's shift is the change in wavelength from excitatory photons to emitted photons [Valeur (2001)]. NBIPS' orange wavelength fluorescence is not usable as an input to subsequent gates, as shown in Figure 7.11, this being a critical limitation of our approach.

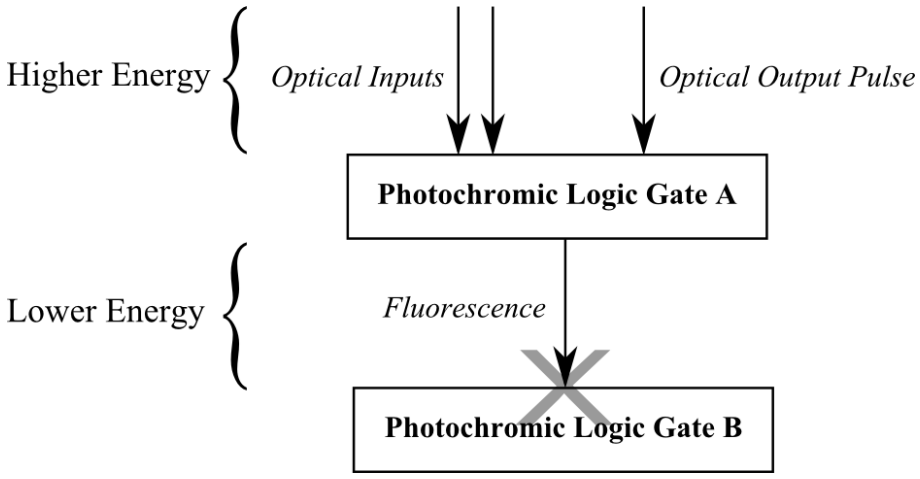



*Figure 7.11: The direct outputs of photochromic logic gates are not suitable for direct use as inputs into another logic gate. eNBIPS logic gates and most fluorescent compounds are subject to Stoke's Shift; emitted photons will be of a lower energy than the excitatory photons.*

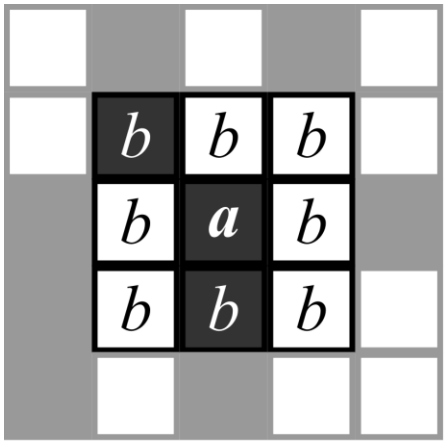
This limitation can be circumvented by using an emission detector which converts fluorescence into a control signal for a light source, which is the approach our experimental setup uses. However, for future work involving synthetic biology where NitroBIPS or other spiropyrans are introduced into biological systems, the output of a photochromic logic gate need not be fluorescence but rather the effect the photochromic molecule has on the tagged process. For example, spiropyrans can be introduced into lipid membranes, and cause perturbation and leakiness in that membrane when in the MC state [Ohya et al. (1998)]. In this case, the output of a logic gate is not fluorescent but rather the release of the membrane contents, which could be used as inputs to other molecular switch logic gates (though not photochromic logic gates). This is discussed in more detail in *Chapter 9: Discussion and Conclusions.*

# Chapter 8: Photochromic Molecular Elementary Cellular Automata

The Turing machine was Turing's formalised description of the simplest universal computer, expressed as a mechanical metaphor. By comparison, elementary cellular automata (ECA) are an abstract simple universal computer proven equivalent to a Turing machine [Cook (2004)]. An ECA is a specialised form of the more general cellular automata (CA). In this section we define photochromic ECAs, demonstrate their implementation and discuss the possibilities this affords us.

## 8.1. Introduction

A cellular automaton is a discrete model often used for modelling physical systems, consisting of a regular array of cells - each possessing a discrete state from an alphabet - where the states of all cells are updated in parallel each time step in accordance with a transition function and the state of the neighbouring cells.




Transition Function

| Number of Neighbours 'Alive' | Current Cell State | New Cell State |
|---|---|---|
| <2 | Either | Dead |
| 2 or 3 | Alive | Alive |
| 3 | Dead | Alive |
| >3 | Either | Dead |

*a*: Current Cell.
*b*: Neighbours (Moore neighborhood)

*Figure 8.1: An example two-dimensional cellular automata cell being updated in accordance with a transition function. The transition function here is Conway's Game of Life [Gardner (1970)]. With two live neighbours, cell* a *remains alive.*

One of the earliest examples of a cellular automaton was von Neumann's universal constructor; a pattern in a two dimensional CA with 29 states that was capable of self-replication [von Neumann (1966)]. Conway's Game of Life [Gardner (1970)] was a much simpler example of a CA, abstracting biological life with only two states (alive and dead, see Figure 8.1), yet it is capable of universal computation [Berlekamp et al. (1982)].

Wolfram defined the ECA as the simplest possible CA [Wolfram (1983)] [Wolfram (2002)]; a one-dimensional array of two-state cells with a neighbourhood consisting of the cell and its two immediate neighbours. However, even this most basic representation has been proven to be universal [Cook (2004)], and hence is one of the simplest universal computational models.

We sought an implementation of an ECA for two reasons. Firstly it would represent a third universal computing paradigm implemented with eNBIPS. Secondly the simplicity of the transition functions mean they can be expressed as a logic circuit allowing us to calculate a cell's next state in the cell itself without simply checking a lookup table, demonstrating the dynamic reassignment of eNBIPS from data storage to data processing. An ECA shares many features with a Turing machine, including a one-dimensional array of cells, cells storing a set of discrete

states, and a transition function to update the state, allowing us to extend the Turing machine's functionality.

## 8.2. Implementation of Photochromic Elementary Cellular Automata

An elementary cellular automaton is a one-dimensional array of cells *C*, where each cell stores a binary state 0 or 1. An initial condition defines the state each cell $C_i$ begins in, defaulting to 0. Each cell is updated every generation by a transition function *F* dependent on the state of $C_i$ and the state of the two cells to the immediate left and right; $C_{i-1}$ and $C_{i+1}$. This transition function can be expressed as. $\forall i, C_i = F(C_{i-1}, C_i, C_{i+1})$ Every generation, the entire array of cells is updated in parallel until a generation limit $g_{max}$ is reached. The ECA has no other halting condition.

Unlike Turing machines, the cell array C is not necessarily limitless, but instead has one of three possible boundary conditions. The first possibility is for the cells immediately beyond the boundary of the array (non-cells) to be considered as a fixed value (typically 0). For the second possibility the array wraps, such that $C_{|C|} = C_0$ and $C_{-1} = C_{|C|-1}$. The third possibility is for the array to be limitless, and automatically adds size as the pattern grows.

Whereas a Turing machine has defined mechanical components that must be implemented in a photochromic system, the abstract nature of ECAs means we have greater freedom in the manner in which components are implemented. These are expressed in similar terms to Turing machine components, and consist of:

1. *Right and Left Arrays:* Similar to the tape in the photochromic Turing machine, the right and left arrays are both on a single eNBIPS sample and consist of two one-dimensional arrays of photochromic binary registers representing the cells of the array C. As our hardware setup cannot address multiple registers simultaneously - and hence cannot update all cells in parallel - we require two arrays $C_R$ and $C_L$ to store the results of each cell update, as shown in Figure 8.2. The first generation and all subsequent odd generations will apply the transition function to every cell in $C_R$, but write the results to $C_L$. Even generations will write from $C_L$ to $C_R$.

As samples cannot be limitless, we can use either the first or second boundary condition. We chose the fixed non-cell choice, where cells $C_{-1}$ and $C_{|C|}$ are always 0, as shown in Figure 8.3. The maximum size of the arrays is determined by the sample; if the sample S can store |S| registers, then the arrays are $\left\lfloor {|S|}/{2} \right\rfloor$ long.

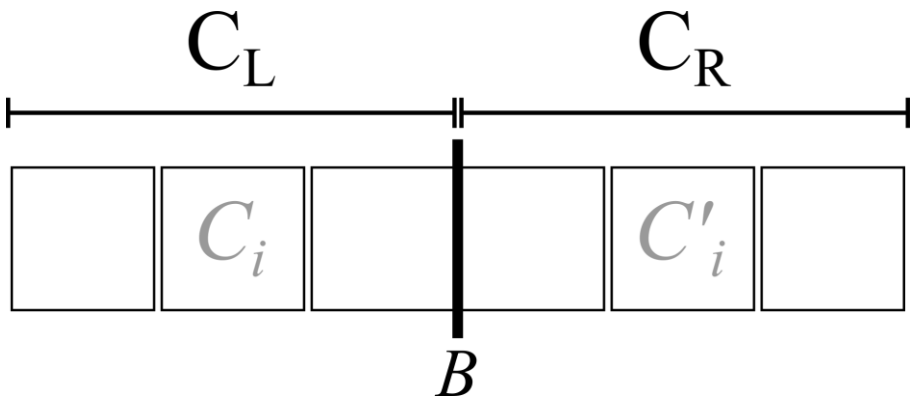



*Figure 8.2: Example ECA arrays with |S|=6. Virtual boundary* B *divides the arrays. Example cells* $C_i$ *and* $C'_i$ *show how cells correspond across arrays.*


*Figure 8.3: Addressing an ECA cell beyond the array boundary. Two virtual 'non-cells' (grey) surround the array, and are always state 0.*

2. *Head:* As with the Turing machine, the head is movable illumination allowing us to address individual cells. This is necessary due to the limitation of our system not operating in parallel.
3. *Transition Function*: Whereas the transition function for Turing machines is a lookup table instructing the machine on its next command, the transition functions for ECAs is expressible either as a table, as a logical function or as a logic circuit, for an example see Figure 8.4. The function *F* applied to $C_i$ takes

the states for cells $C_{i-1}$, $C_i$ and $C_{i+1}$ as inputs, and returns $C_i'$. We implement the function as a logic circuit which runs on $C_i$.

Implementation requires four atomic actions. These actions and their implementations are:

1. *Move*: Though the definition of ECAs requires no head and no movable addressing, our system can only access a single cell at a time. As a result the implementation requires movable illumination. The move action moves the illumination left or right along the array or over the virtual boundary between $C_L$ and $C_R$ arrays. Unlike Turing machines, Move is not restricted to a single cell movement, but could be many.
2. *Observe*: Reading the cell the head is currently addressing. This is equivalent to reading a register. Decrement reading with resetting is used when cells are read (see *Chapter 5: Photochromic Molecular Registers*) as the cell will be read multiple times.
3. *Print*: Altering the value of the cell to 0 or 1 in accordance with the transition function result. This is equivalent to writing to a register.
4. *Execute*: For each generation, for each cell, an execute action reads the state of the cell's neighbours, runs the logic circuit, and prints the result to the corresponding cell in the opposite array. A flowchart of Execute's operation is shown in Figure 8.5.

These actions are combined to execute the ECA, as shown in Figure 8.6.

*a)*

| $C_{i-1}$ | $C_i$ | $C_{i+1}$ | $C'_i$ |
|---|---|---|---|
| F | F | F | F |
| F | F | **T** | **T** |
| F | **T** | F | **T** |
| F | **T** | **T** | **T** |
| **T** | F | F | F |
| **T** | F | **T** | **T** |
| **T** | **T** | F | **T** |
| **T** | **T** | T | F |

*b)* $C'_i$ = ($C_i$ AND (NOT $C_{i-1}$)) OR ($C_i$ XOR $C_{i+1}$)

*c)*

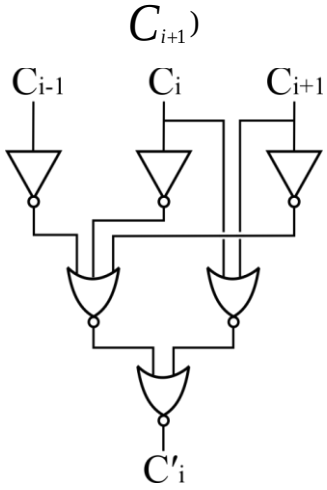



*Figure 8.4: Representations of the transition function for Rule 110. a) As a table. b) As a logic function. c) As a logic circuit optimised for photochromic logic gates.*

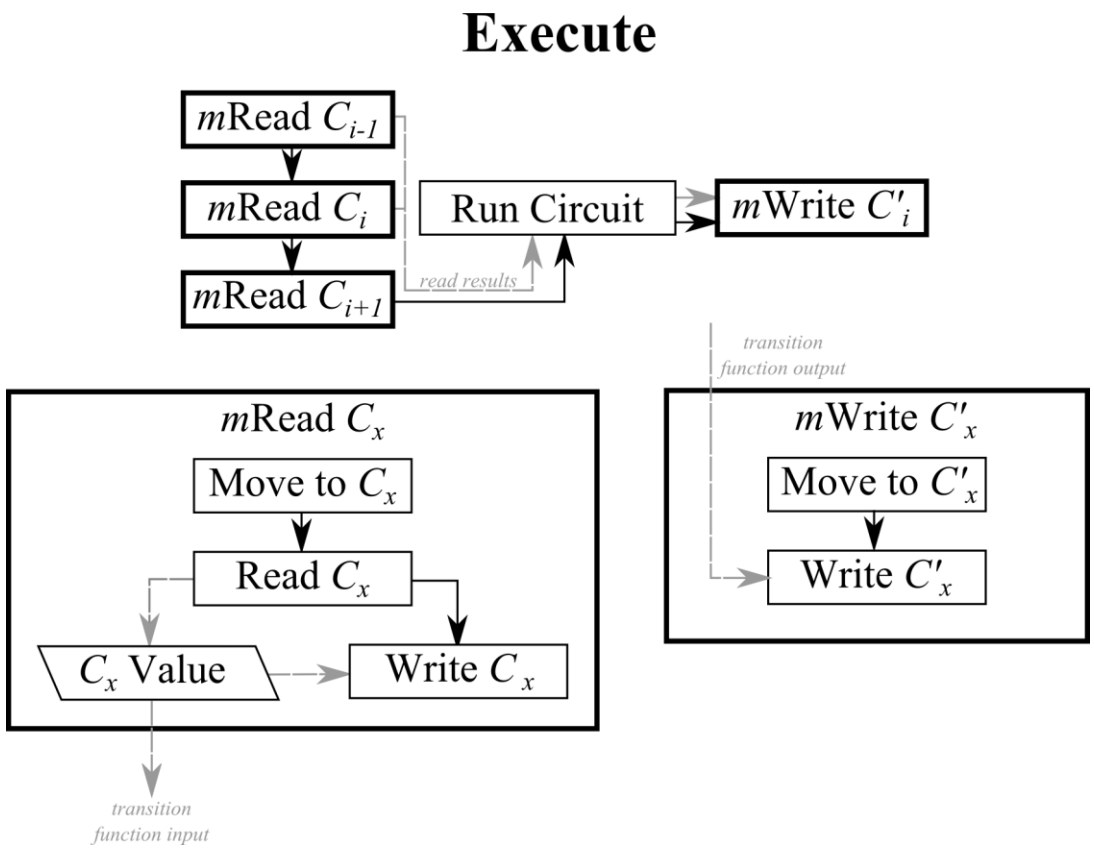



*Figure 8.5: Flowchart of the photochromic ECA execute action.* m*Read and* m*Write are compound actions that move to a cell and read/write it, shown underneath.*

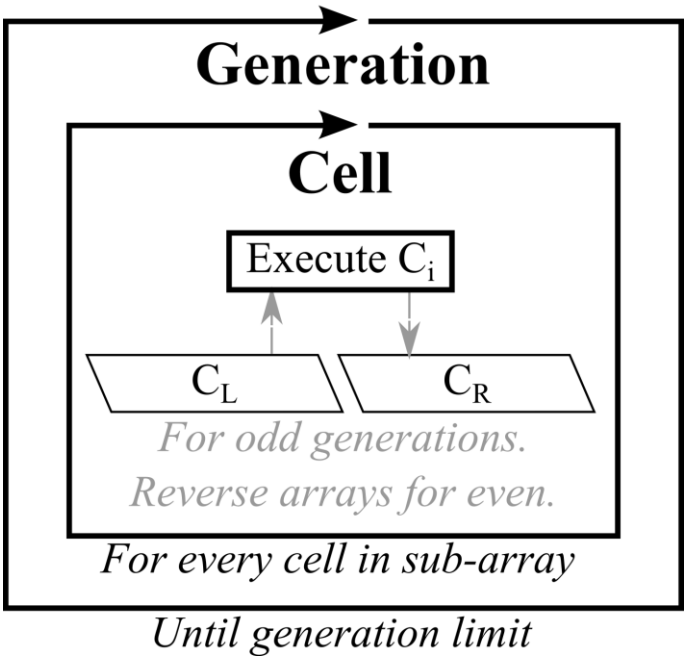



*Figure 8.6: Flowchart of system operation for photochromic ECAs. Every cell is executed once for each generation, and the ECA executes until the generation limit is reached.*

## 8.3. Execution of Photochromic Elementary Cellular Automata

We sought to implement one ECA from each of Wolfram's Wolfram Classes (See *2.2. Unconventional Computing*). The following figures (Figure 8.7 to Figure 8.11) show the executions of five example ECAs covering the range from class 1 to class 4. Each figure has five components. a) is a brief summary of general ECA operation. b) is a table representation of the transition function, and c) is that transition function as a logic circuit. d) are the results of the ECA execution, 10 generations with an array size of 5. Each iteration consists of two rows; the first is the graphical representation of the cell states and the two virtual non-cells in grey, and the second consists of sets of three waveforms for each cell; the readings of $C_{i-1}$, $C_i$ and $C_{i+1}$ for each cell in the array. To the left is a column indicating the generation, and to the right is which array is in use. e) is a comparison of the expected result and the result given by the execution. In each case, the execution of the ECA produced the expected result.

Similarly to photochromic tape Turing machines from the previous section, the duration of colouration stimulus to set a cell to $V_1$ is longer than would be used for a simple register. Though the time between cell reads is now fixed per ECA

rather than indeterminate, the time a cell must remain in its value is high and hence subject to thermal relaxation, placing a limit on the width of the ECA. The time to execute each ECA iteration varied depending on the complexity of the transition function, approximately 30 seconds per cell read, 30 seconds per gate and 1 minute per cell write (i.e. as long as 27 minutes per iteration for rule 110). The ECA width of 5 is the highest width for which Rule 110 can write $V_1$ to $C_0$, and for it to still be read as $V_1$ during the next iteration. Though simpler rules could be executed with more cells due to the reduced circuit execution time, a width of 5 is sufficient for a proof of concept and is kept for consistency.

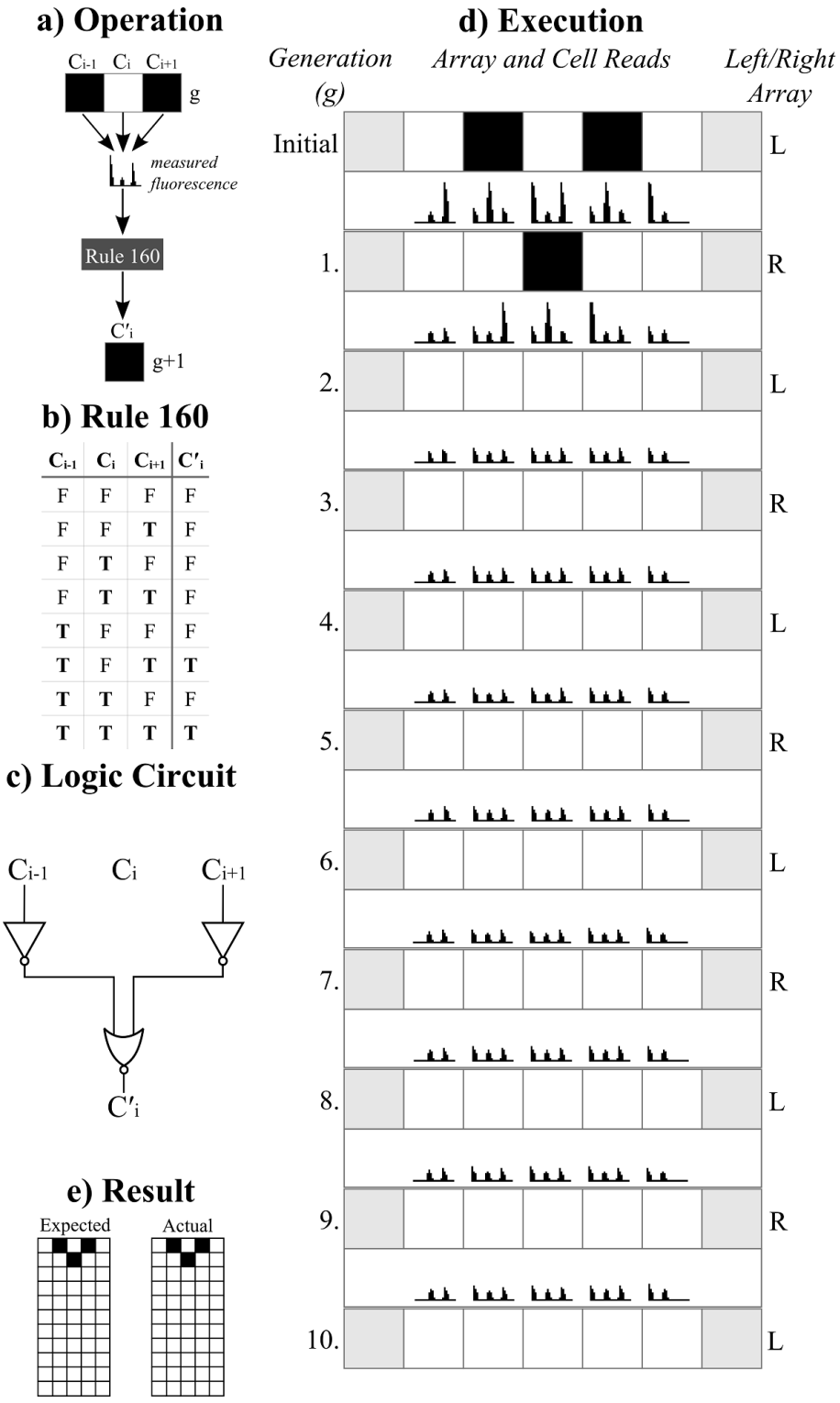



*Figure 8.7: Execution of Rule 160. Rule 160 is a class 1 ECA, and rapidly converges to a uniform state of 'off' cells. The initial condition was two 'on' cells.*

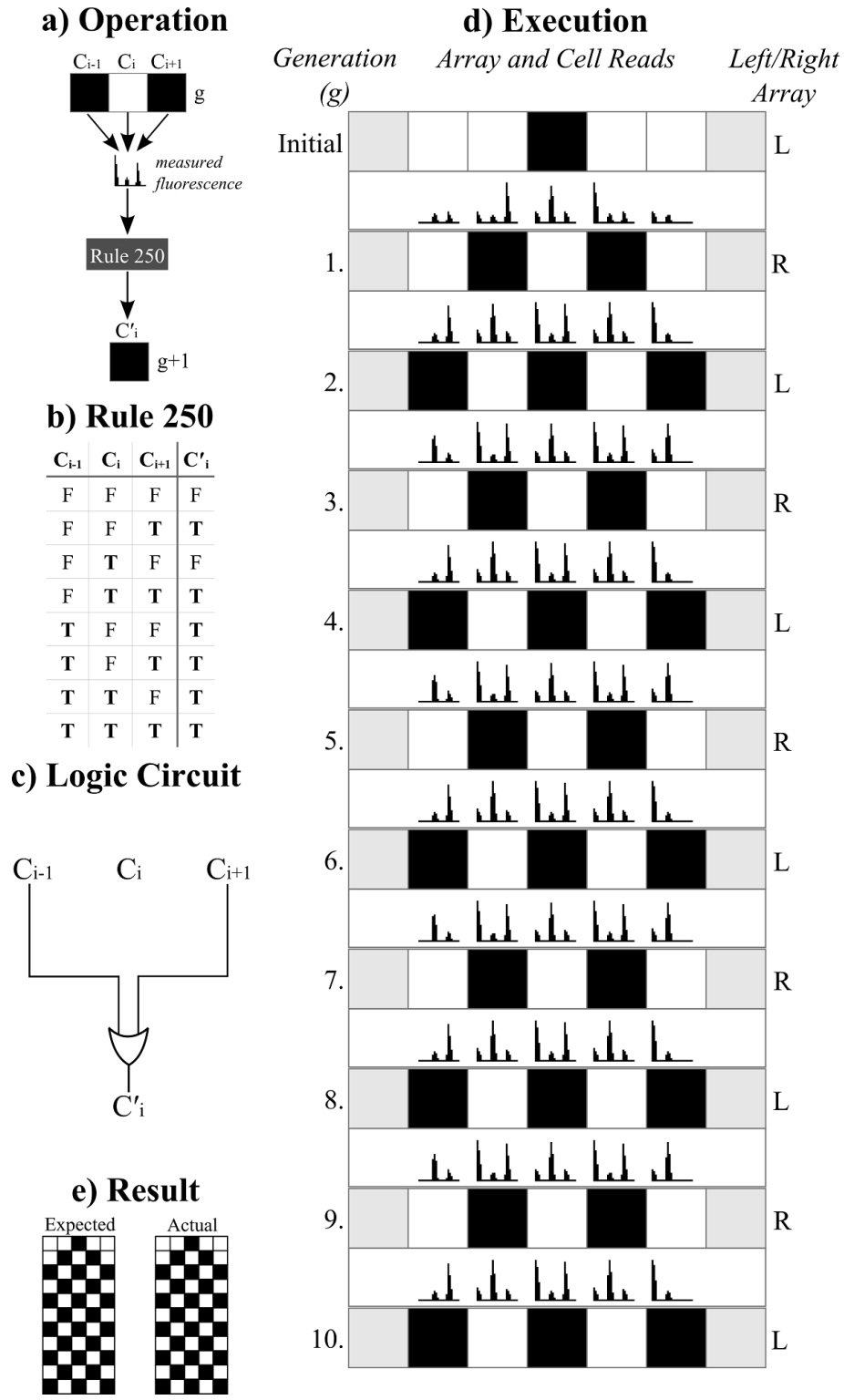



*Figure 8.8: Execution of Rule 250. Rule 250 is a class 1 ECA, though the fixed values of the boundary cells causes it to form a repeating alternating pattern. This is a limitation of the chosen boundary condition. The initial condition was a single on cell.*

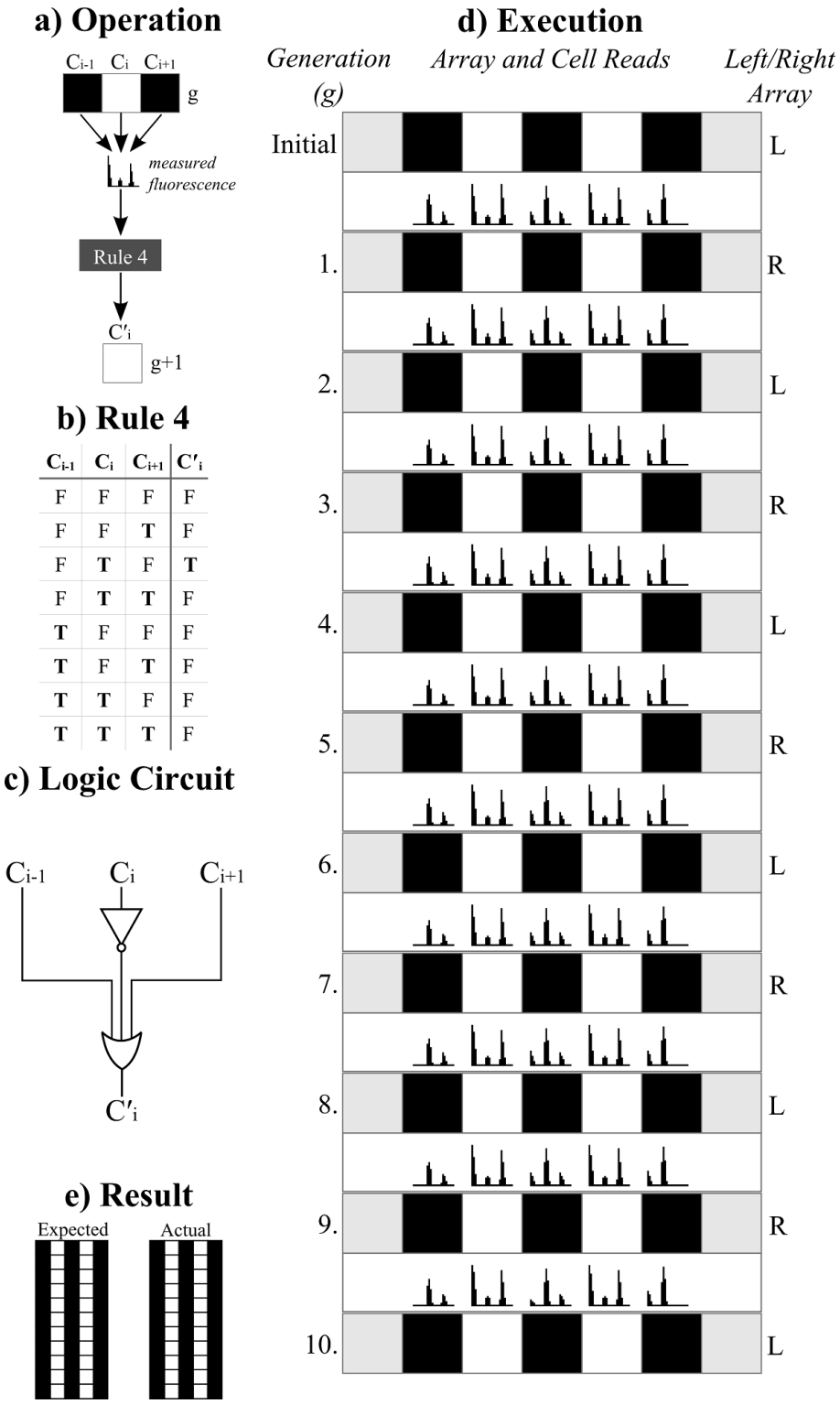



*Figure 8.9: Execution of Rule 4. Rule 4 is a class 2 ECA, forming repetitive structures. The initial condition is alternating cell states.*

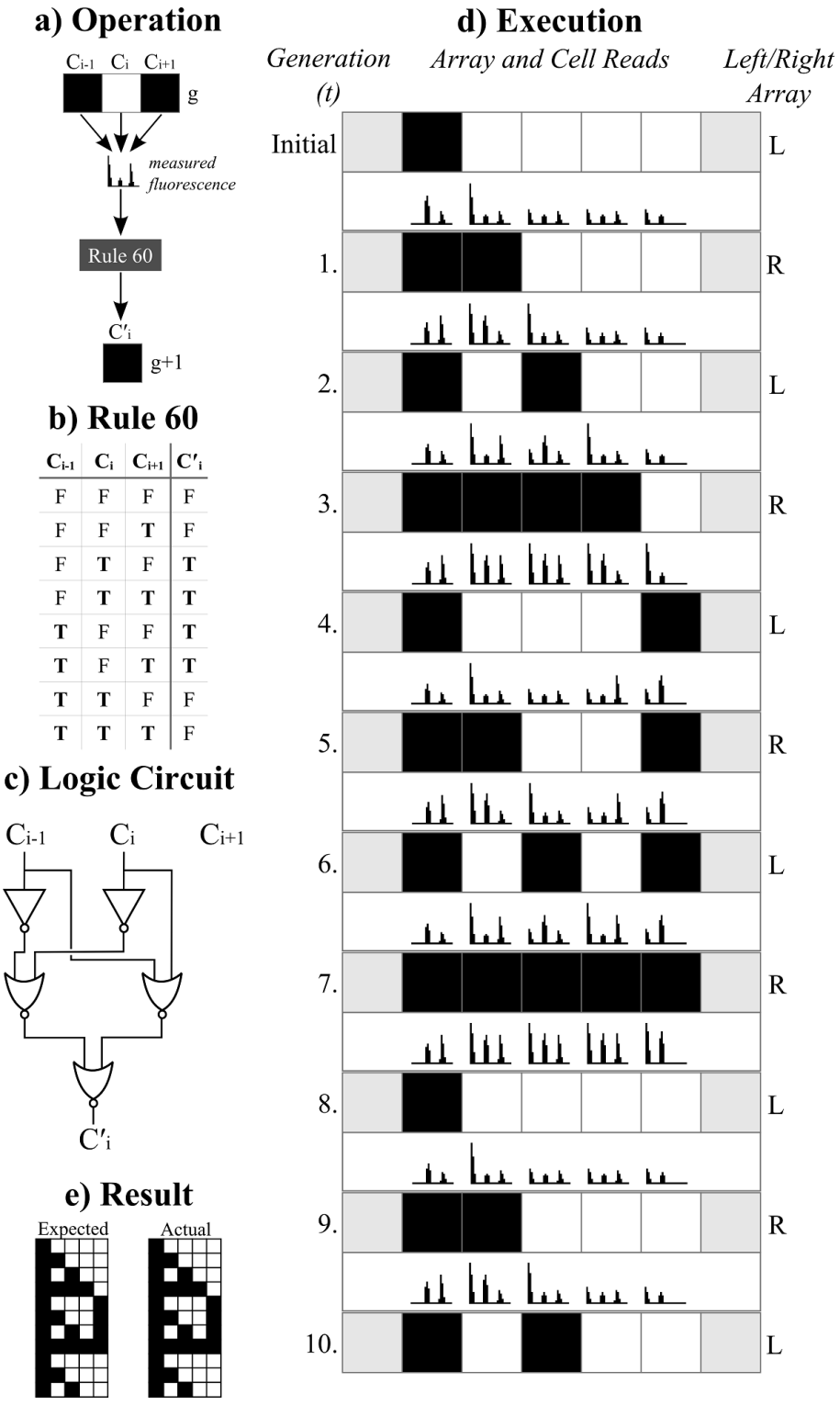



*Figure 8.10: Execution of Rule 60. Rule 60 is a class 3 ECA, forming random structures. The initial condition is $C_0 = 1$.*

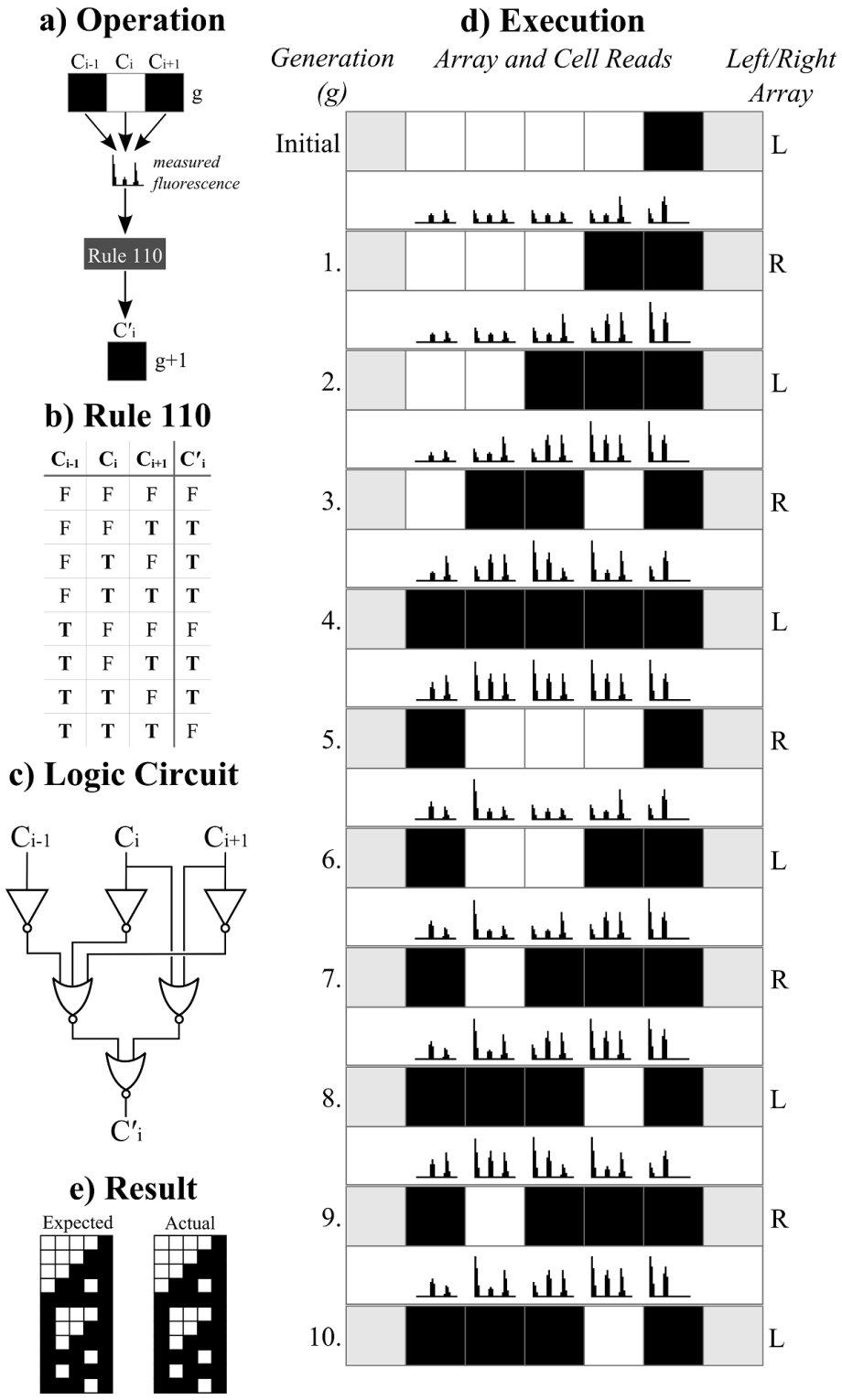



*Figure 8.11: Execution of Rule 110. Rule 110 is a class 4 ECA, and is proven to be capable of emulating a Turing machine, and hence of universal computation. [Cook (2004)] The initial condition is $C_5 = 1$.*

## 8.4. Discussion

In this section we have demonstrated that it is possible to implement photochromic ECAs and have given examples of all four Wolfram classes of ECA including the universal Rule 110. ECAs have been implemented in a way to suit our hardware setup and to showcase the capabilities of photochromic computation. In its current form the hardware can only read or write to a single cell register at a time, requiring two parallel arrays of cells, and executing each cell in turn per generation. However, a logic circuit implementing the transition function is executed within the cell register leaving it in the new state demonstrating the ability for photochromic devices to have their operation redefined dynamically. It is possible to parallelise photochromic ECAs with the hardware modifications discussed in *Chapter 9: Discussion and Conclusions*.

In the current hardware, the size of the arrays used for ECA execution is limited not by the illumination size and traverse of the stepper motor but by the length of time the cell register will hold its value before thermal relaxation causes it to revert to zero. The more complex the logic circuit needed to execute the transition function, the longer it takes to update each cell. For example, this effect can be seen in Figure 8.11; the peak fluorescence measured for $C_0$ during the execution of $C_0$ for all generations >4 during Rule 110 is very close to background noise, as the register is set but unread for a long period. As a result, an execution of Rule 110 with 5 cells has been shown to fail due to a cell thermally relaxing while the operating temperature is higher than normal, resulting in a cell which was set to true being read by the subsequent generation as false. However, Rule 110 was the most complex rule tested, so every rule with a lower execution time was reliable once correctly calibrated. This is a limitation specific to photochromic devices as discussed in previous sections, and could be overcome with the use of a more thermally stable spiropyran, cooling of the system or by increasing the speed of execution.

Extension to 2D or even 3D cellular automata is possible, provided the thermal decay issues are handled. Upgrading to a 2D spatial light modulator or use of two-photon excitation allow for the control of illumination in more dimensions and the transition functions could be implemented as a logic circuit with more inputs. The logic circuits for two and three dimensional CAs would be

correspondingly more complex and time consuming to execute, reiterating the need for thermal decay to be resolved or put to use somehow.

The implementation of ECAs is another example of a universal computing paradigm and the advantages of movable illumination to independently control sub regions of a homogenous eNBIPS sample. It also shows the capability for different functions (i.e. registers, logic circuits etc) to be executed on a spiropyran-doped substrate without invasive modification.

# Chapter 9: Discussion and Conclusions

## 9.1. Overview and Contributions

This thesis addressed the following research goals, as laid out in *Chapter 1: Introduction*:

- Investigate the properties and capabilities of a biocompatible molecular switch which could be used to implement a register.
- Implement registers using these molecules, and implement fundamental conventional logic elements as an extension to these registers.
- Extend the fundamental conventional logic elements to implement a universal computing paradigm.

Goal one is addressed in *Chapter 3: Computational Model and Programmability* and *Chapter 4: Methods* where methods and protocols for conducting experiments on spiropyrans were defined. NitroBIPS was encapsulated in a polymer matrix, providing a safe and easy to use test bed for the characterisation of the molecule. An optical system and control mechanism using LabVIEW programs are documented for exposing the molecule to stimulus and recording fluorescence.

The system is sufficient for the proof of concept work detailed in this thesis but could be improved, as detailed in *9.2 Future work*. Using this set up, the molecule's

behaviour and properties are characterised for use in completing the subsequent goals.

Goal two is addressed in *Chapter 5: Photochromic Molecular Registers* and *Chapter 7: Photochromic Molecular Logic Gates and Logic Circuits*Photochromic Molecular Logic Gates. Photochromic registers are the simplest core component from which the other components are drawn. The ability to store data is an important part of any computing paradigm and we sought a method implementable with NitroBIPS and extendable to other molecular species. The ability to store data as the relative proportion of two molecular states is applicable to many molecular switches, and NitroBIPS lends itself well with a simple way to measure the relative states via fluorescence.

One defining feature of photochromic registers compared to conventional data stores is that reading a photochromic register also alters its state. As the excitation and decolouration absorption spectra for NitroBIPS are the same, any excitation also causes decolouration. We discuss three methods to address this issue in *Chapter 5: Photochromic Molecular Registers,* but as shown in *Chapter 6: Steps Toward Photochromic Molecular Turing Machines*, sometimes this issue is not problematic as any cell that is read will have a new value written anyway.

Another defining feature of photochromic registers is that the number of values it can store is user defined. This thesis has typically considered registers with a low number of values; to ensure reads are robust both to noise and to thermal relaxation. Improvements to reliability could be made with the introduction of error checking codes, though standard methods may not be applicable due to the destructive nature of read operations.

High reliability value encoding is not the only option. More values can be defined with a lower certainty all the way to the logical conclusion of continuous values, where the number of values is determined by the resolution of the detector. For synthetic biology applications, the abstraction to values may no longer be applicable where the form of the molecule directly affects biological processes.

*Chapter 6: Steps Toward Photochromic Molecular Turing Machines* demonstrates an implementation of the tape in the classic model of universal

computing; the Turing machine. With movable illumination on a single axis built into our hardware setup, the ability to implement a one-dimensional array of parallel registers was reminiscent of the one-dimensional tape of symbols that represents the core of a Turing machine. A photochromic Turing machine tape utilises registers to demonstrate how eNBIPS can store data within a larger universal.

Photochromic Turing machines show the potential for eNBIPS for medium term data storage, by having different cells read and written at indeterminate lengths of time. The rate of thermal relaxation on eNBIPS samples has limited this proof of concept to binary cells, demonstrating the limitations of this approach.

Photochromic Turing machines also show the first steps to downloading all the data necessary for the Turing machine execution onto the eNBIPS sample itself, by storing the I-state on tape and demonstrating how, in principle, the transition function could be stored too. At present, the I-state is stored on the same axis as the tape. An additional dimension to addressability would allow for the I-state and transition function to be stored independently of the tape.

Continuing with the theme of low-range high-reliability registers, logic gates extend registers; using just two or three values. *Chapter 7: Photochromic Molecular Logic Gates and Logic Circuits* utilises the same hardware, same eNBIPS samples and same concepts as registers. By exposing samples to pulses of light, the NitroBIPS samples are left in a fluorescent or non-fluorescent state representing the output of the gate. By using different pulse sequences almost all common logic gates can be implemented. These logic gates include many universal sets, including the sole sufficient operators NOR and NAND.

Logic gates use exactly the same hardware as registers, potentially offering the system the ability to prioritise execution speed or storage capacity dynamically. eNBIPS samples can be repurposed non-invasively; the same substrate and sub-regions of that substrate can execute almost any logic gate and store data. *Chapter 8: Photochromic Molecular Elementary Cellular Automata* demonstrates how a cell can both store a state then also execute logic functions in the same sub-region to leave that cell in a new state.

Logic gates require a temporally ordered set of inputs to operate, a potential limitation depending on their use. However, the capability for some gates (such a n-NOR) to have both disordered and arbitrary amounts of inputs, coupled with inputs not needing to arrive in parallel may also prove to be an advantage in some implementations where synchronisation is not guaranteed or necessary.

Goal 3 is answered by *Chapter 7: Photochromic Molecular Logic Gates and Logic Circuits* and *Chapter 8: Photochromic Molecular Elementary Cellular Automata*; each implements a universal computing paradigm. Logic Circuits are demonstrated with two possible workflows for circuit implementation. Drag & drop allows the simple placement and connection of LabVIEW subVIs to implement logic circuits. For more complex circuits, circuit compilation makes circuit design easier via Logisim and allows for the automated serialisation and uniquification.

The major issue at this stage with logic circuits is that the output of a logic gate cannot be used as the input to a subsequent gate. The Stoke's shift means that the energy of emitted photons is lower than the excitation photons. Instead, it is necessary to read fluorescence and convert it into a control signal for input pulses. This is currently performed by a controlling computer, but a minimal optical amplifier could be built. Another issue is non-parallelism; circuits must be serialised, though we discuss methods to fix this in *9.2 Future work.*

The second universal computing paradigm is demonstrated in *Chapter 8: Photochromic Molecular Elementary Cellular Automata*, with the transition rules implemented as logic circuits. ECAs represent the culmination of parts from the entire thesis, utilising registers to store the cell values, logic circuits executed at the same site as the registers to execute the transition rule, and multiple cells existing in parallel. A selection of rules spanning the four Wolfram classes were implemented, including the universal rule 110.

ECAs show the ability for eNBIPS devices to operate a number of different functions without invasive alteration of the underlying sample. However, the time required for our system to execute the transition functions limits the size of ECAs possible before cells thermally relax. This limitation is not insurmountable, but is primarily a product of low LED intensities at the sample, the high rate of thermal relaxation of NitroBIPS, and the serial nature of the hardware.

## 9.2. Future work

This thesis has studied many aspects and possibilities of NitroBIPS and photochromic molecules in general and established a solid foundation from which to proceed with synthetic biology applications in future research. This section details some of the more promising directions that have been identified for future work. These include both improving the experimental setup and also making first steps to integrating NBIPS into liposome logic and other synthetic biology.

### *9.2.1 Improved System Performance*

One issue with the current hardware setup is the time necessary to colour/decolour samples, and as a consequence registers undergo significant thermal relaxation. This is an engineering issue and is not insurmountable. One such method would be photochromic discs; spiropyrans can be introduced into rigid polymer matrices (e.g. photo-reactive ophthalmic lenses [Le Naour-Sene (1981)]). By casting these samples into discs, it would be possible to address samples similar to a modern DVD or Blu-Ray player.

For example, the Blu-Ray standard uses a 580nm spot size, addressed with a 120mW gallium nitride laser diode. [Sony (2012)]. Laser diodes with wavelengths suitable for eNBIPS samples are available (e.g. ThorLabs DJ532-40 40mW 532nm Laser Diode and ThorLabs L375P020MLD 20mW 375nm Laser Diode), though the higher wavelength of the 532nm laser diode would increase the spot size; the ability for a lens to resolve detail (the resolving power) is proportional to $\lambda/2N_{A.}$ Blu-Ray players also feature a photodiode for reading reflected light. A similar system could be used to read fluorescence signals.

A disc-based system or other improved hardware setup would allow for a greater data density through a greater resolution of parallel registers. Execution of logic gates would be quicker as shorter pulses of light would be required to read/write gates. Thermal relaxation would be less of a problem, both because the rigid polymers that have been indentified cause spiropyrans to have very low rates of thermal relaxation [Dvornikov et al. (1994)] and because data would be accessed quicker.

### *9.2.2 Parallelisation*

One of the more obvious ways the experimental setup could be extended is to improve the parallelisation possibilities of the system. At present, a sample of eNBIPS can have sub-regions illuminated, but only one at a time and only on a one dimensional axis. As a result, an eNBIPS sample can store multiple values as registers simultaneously, but cannot read/write registers nor execute logic gates in parallel. An ECA also cannot update all cells in parallel.

One solution to this is to use a Spatial Light Modulator (SLM). An SLM is an array of micro-mirrors or liquid crystal reflectors which alter light polarisation selectively, allowing a polarisation-filtered pattern of light to be reflected onto the sample. SLMs are the mechanism via which modern digital projectors operate. The use of an SLM would allow the selective addressing of regions on a sample of eNBIPS up to the size of the SLM (typically display resolutions, SXGA, WXGA etc). SLMs can operate at the desired wavelengths (e.g. they can be used for UV microstereophotolithography at 351nm [Farsari et al. (1999)])

Illumination could be directed in parallel spatially. For fluorescence to be monitored in parallel, the photodiode would be replaced with a camera, allowing multiple locations to be monitored simultaneously. The implementation of an SLM and camera and an associated LabVIEW control program would be sufficient to implement parallel functionality. This would allow for non-serialised logic circuit to be executed, multiple registers written to and read simultaneously, and the implementation of two-dimensional cellular automata. The principles could also be applied to addressing spiropyrans *in vitro*, exposing only selected regions to stimulus.

### *9.2.3 Three dimensional Addressing*

Though the techniques in *9.2.2* are suitable for two-dimensional addressing of eNBIPS sub-regions, it is possible in principle to expand to three-dimensional addressing of sub-regions in eNBIPS blocks by using two-photon excitation. Two-photon excitation is a process where the energy of a photon required to cause a state change is instead replaced by two photons with half the energy (i.e. double the wavelength). Only molecules absorbing two photons simultaneously can change state and this is only possible if the photon density is extremely high. This is

achieved with a very high intensity and very brief laser pulse focused to a small spot. As the photon density is only high enough as the focussed spot, state transitions are limited to the target voxel.

Two photon excitation has been applied to molecular switches for data storage purposes [Belfield et al. (2002)] [Dvornikov et al. (2009)] and could be applied to eNBIPS if light sources of appropriate wavelengths could be acquired. It would allow for a higher density of registers and logic gates in a volume, and for three-dimensional cellular automata to be implemented. The same principles could be applied to spiropyrans *in vitro*, allowing the switching of molecules selectively in three dimensions.

### *9.2.4 Introduction of Techniques to Synthetic Biology*

Photochromic logic gates have potential applications within a synthetic biology framework. As discussed in *Chapter 2: Background Material*, NBIPS and other molecular switches can potentially moderate biological processes, including protein binding [Sakata et al. (2005)] and membrane leakiness [Ohya et al. (1998)]. Spiropyrans can be encapsulated within liposomes [Carol A. Jennings et al. (1997)] and liposomes can implement logical functions [Smaldon et al. (2010)].

Photochromic registers applied to synthetic biology would allow for graduated responses, with each value corresponding to an increase or decrease in membrane leakiness, protein binding etc. Figure 9.1 shows a theoretical example of this.

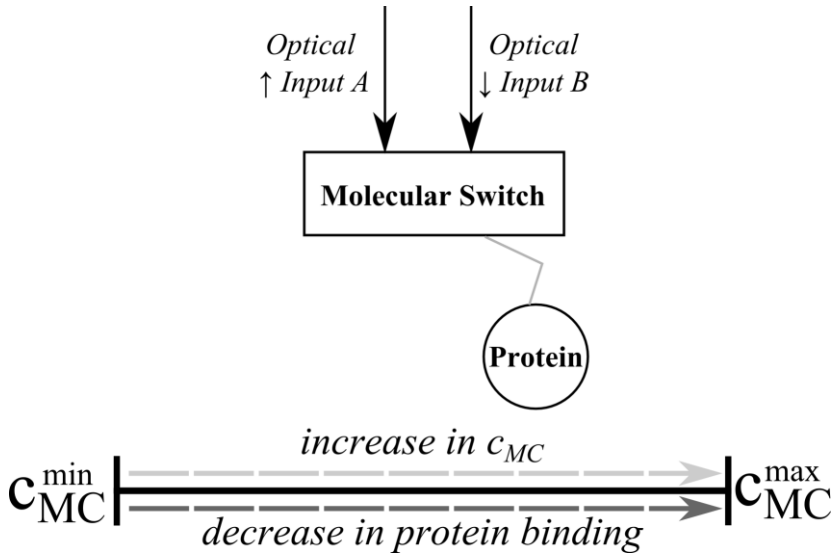



*Figure 9.1: A photochromic molecule with optical inputs inhibits protein interactions proportional to the value of the register.*

Photochromic logic gates require output pulses so the operator can determine the final truth value of the gate. If the gate instead did not have an output pulse, it would remain in the final value. If the NBIPS were attached to a protein or a membrane, the final value would represent the presence or absence of protein binding, or membrane perturbation.

Figure 9.2 shows how a molecular switch with a high thermal equilibrium (due to body temperature or similar) has a high proportion of molecules in the unfolded state, inhibiting protein interaction. The presence of chemical or optical inputs will fold the molecular switches and permit full protein interaction. This is equivalent to a 2-NOR gate. If these inputs are removed, thermal effects will unfold the molecules and inhibit protein interaction again.

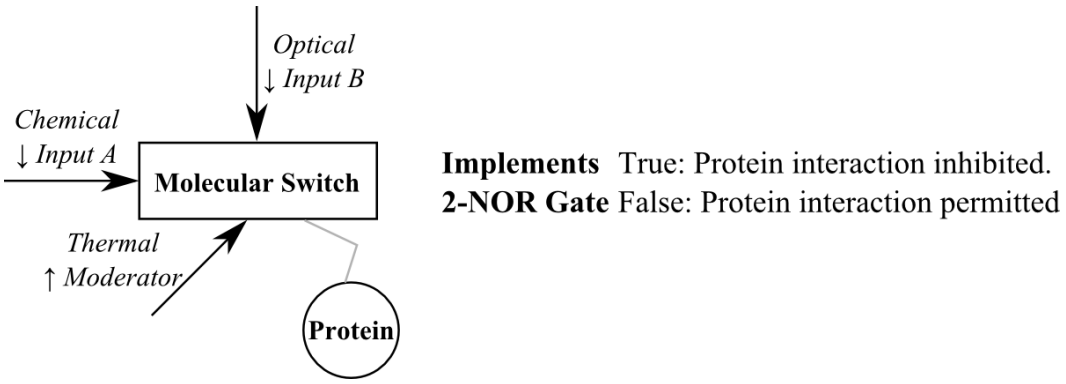



*Figure 9.2: Example application of a molecular switch logic gate for inhibiting protein binding.*

Figure 9.3 shows another theoretical example of an application of a molecular switch logic gate. If the inputs are intense enough to fully colourise the molecular switch, the presence of one or more of the inputs will cause the lipid membrane to become leaky. This is equivalent to the alternative definition of an OR gate. If the inputs are not sufficient to fully colourise the molecular switch, each present input will increase the leakiness of the lipid membrane. This is equivalent to the standard definition of an OR gate.

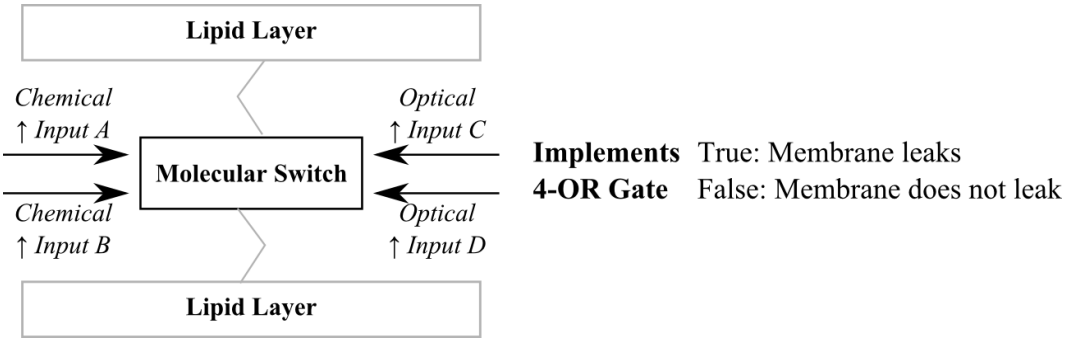



*Figure 9.3: Example application of a molecular switch logic gate for membrane perturbation. A theoretical molecular switch with low quantum yields (and hence negligible thermal transitions) with two colouring chemical and two colouring optical inputs.*

Figure 9.4 shows how a third theoretical molecular switch with a colourising chemical stimulus input is combined with a biological process forming a 1-ID gate. When exposed to an optical excitation pulse (visible light for NBIPS), the molecule

will fluoresce if the chemical input is present (execution path *i*). If there is no chemical input, the molecular switch will not fluoresce during the output pulse (execution path *ii*). This allows molecular switches to be used to report on internal biological processes.

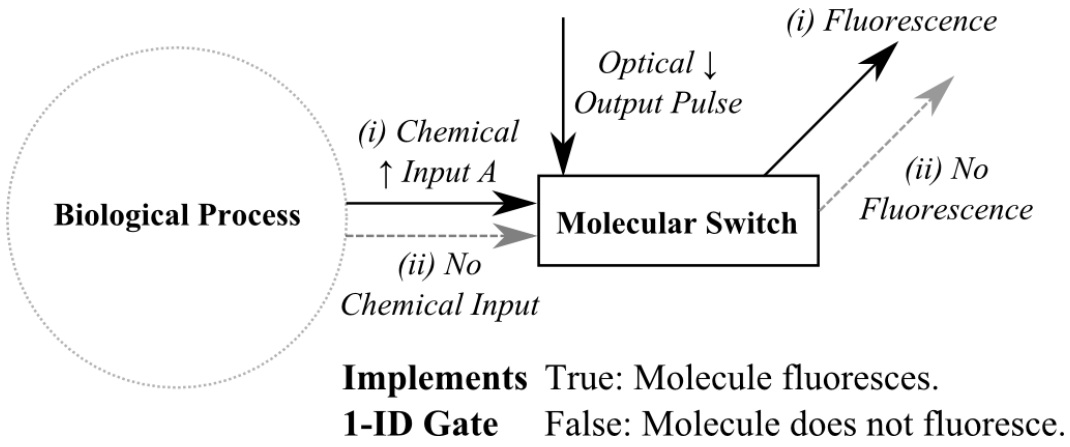



*Figure 9.4: Example application of a molecular switch logic gate for process reporting.*

Photochromic logic circuits have potential applications within synthetic biology, liposome logic and membrane computing. Though the fluorescence of NBIPS is not suitable as an input to another logic gate, the effect of the state of NBIPS might be. As discussed in the previous section, NBIPS can alter biological processes. The result of these changes could be the inputs to subsequent gates, though it's unlikely these would be photochromic. Instead, photochromic gates could act as the I/O system of a larger biological logic circuit, as shown in Figure 9.5. Photochromic logic gates could receive external input in the form of light pulses, and act as the inputs to internal molecular logic circuits. A second photochromic logic gate could then report the result of the circuit via fluorescence. As light is minimally invasive, this allows for input and output without disruption to the system.

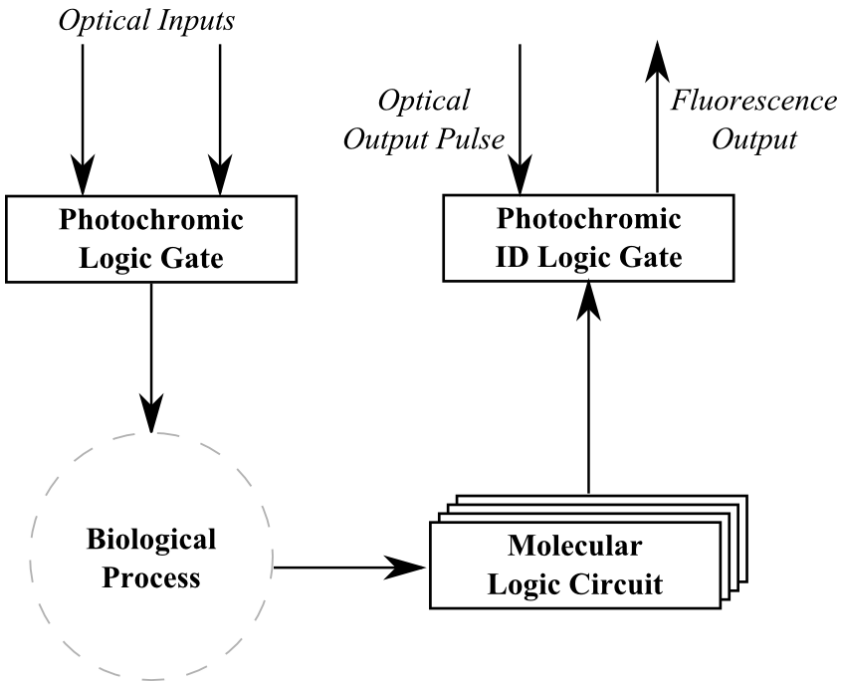
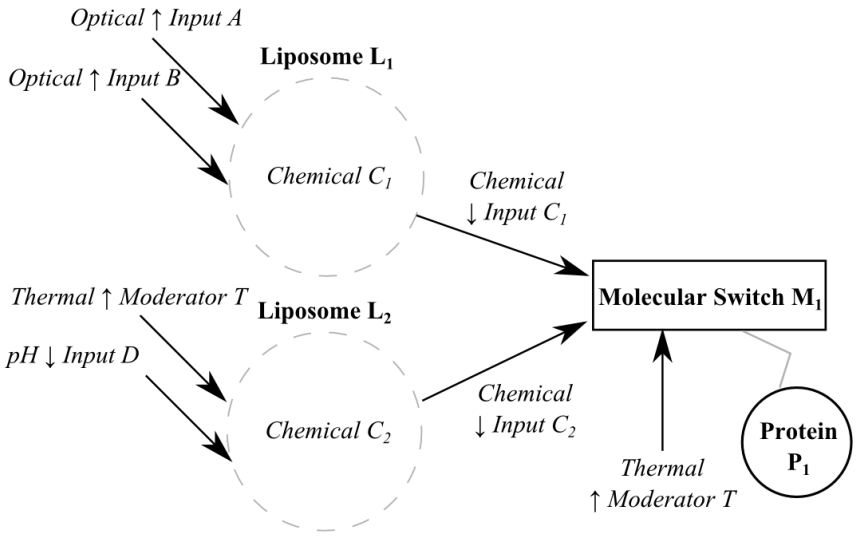



*Figure 9.5: Photochromic logic gates form the input/output path for a separate molecular logic circuit.*

The techniques shown this thesis could be applied to circuits composed of gates implemented with other forms of molecular switch. A potential example is shown in Figure 9.6. The circuit is at body temperature, causing positive thermal moderation to some molecules (*T*). $L_1$ is photosensitive with a low quantum yield. Two positive optical inputs *A* and *B* are required for $C_1$ to leak in meaningful quantities. $L_2$ is pH sensitive; the presence of acid is a negative stimulus. Positive thermal moderation allows a meaningful amount of $C_2$ to leak, but *D* will inhibit this. $M_1$ is bound to $P_1$. Positive thermal moderation means $M_1$ is open and inhibits $P_1$ binding. If either $C_1$ or $C_2$ are present, $M_1$ folds and $P_1$ can bind.




**$L_1$ Implements 2-AND Gate**

A & B: Liposome leaks enough $C_1$ for *Chemical Input* $C_1$ to be *True*.
¬(A & B): *Chemical Input* $C_1$ is *False*.

**$L_2$ Implements 1-NOT Gate**

D: Liposome does not leak $C_2$.
¬D: Liposome leaks enough $C_2$ for *Chemical Input* $C_2$ to be *True*.

**$M_1$ Implements 2-NOR Gate**

$C_1$ || $C_2$: $P_1$ binding inhibited.
¬$C_1$ & ¬$C_2$: $P_1$ binding permitted.

| *A* | *B* | *D* | $C_1$ | $C_2$ | *$P_1$ Binding* |
|---|---|---|---|---|---|
| F | F | F | F | T | **T** |
| F | F | T | F | F | **F** |
| F | T | F | F | T | **T** |
| F | T | T | F | F | **F** |
| T | F | F | F | T | **T** |
| T | F | T | F | F | **F** |
| T | T | F | T | T | **T** |
| T | T | T | T | F | **T** |

*Figure 9.6: A theoretical implementation of a molecular switch logic circuit.*

# References

[Abelson et al. (2000)] Abelson, H., Allen, D., Coore, D., Hanson, C., Homsy, G., Knight, Jr., T. F., Nagpal, R., Rauch, E., Sussman, G. J., Weiss, R., May 2000. Amorphous computing. *Commun. ACM* 43 (5), 74–82.

[Adamatzky (2004)] Adamatzky, A., 2004. Collision-based computing in belousov zhabotinsky medium. *Chaos, Solitons and Fractals* 21 (5), 1259–1264.

[Adamatzky (2007)] Adamatzky, A., DEC 2007. Physarum machines: encapsulating reaction-diffusion to compute spanning tree. *Naturewissenschaften* 94 (12), 975–980.

[Adamatzky et al. (2012)] Adamatzky, A., Holley, J., Dittrich, P., Gorecki, J., Costello, B. D. L., Zauner, K.-P., Bull, L., JUL 2012. On architectures of circuits implemented in simulated Belousov-Zhabotinsky droplets. *Biosystems* 109 (1, SI), 72–77.

[Adleman (1994)] Adleman, L. M., 1994. Molecular computation of solutions to combinatorial problems. *Science* 266, 1021–1024.

[Aizawa et al. (1977)] Aizawa, M., Namba, K., Suzuki, S., 1977. Photo control of enzyme activity of [alpha]-amylase. *Archives of Biochemistry and Biophysics* 180 (1), 41 – 48.

[Amos (2005)]Amos, M., 2005. *Theoretical and Experimental DNA Computation*. Springer, Dordrecht.

[Andersson et al. (2008)] Andersson, J., Li, S., Lincoln, P., Andréasson, J., 2008. Photoswitched dna-binding of a photochromic spiropyran. *Journal of the American Chemical Society* 130 (36), 11836–11837.

[Ando et al. (2007)] Ando, R., Flors, C., Mizuno, H., Hofkens, J., Miyawaki, A., JUN 2007. Highlighted generation of fluorescence signals using simultaneous two-color irradiation on Dronpa mutants. *Biophysical Journal* 92 (12), L97–L99.

[Andréasson et al. (2011)] Andréasson, J., Pischel, U., Straight, S. D., Moore, T. A., Moore, A. L., Gust, D., 2011. All-photonic multifunctional molecular logic device. *Journal of the American Chemical Society* 133 (30), 11641–11648.

[Anelli et al. (1991)] Anelli, P. L., Spencer, N., Stoddart, J. F., 1991. A molecular shuttle. *Journal of the American Chemical Society* 113 (13), 5131–5133.

[Atassi et al. (1995)] Atassi, Y., Delaire, J. A., Nakatani, K., 1995. Coupling between photochromism and second-harmonic generation in spiropyran- and spirooxazine-doped polymer films. *The Journal of Physical Chemistry* 99 (44), 16320–16326.

[Aviram and Ratner (1974)] Aviram, A., Ratner, M. A., 1974. Molecular rectifiers. *Chemical Physics Letters* 29 (2), 277 – 283.

[Bachtold et al. (2001)] Bachtold, A., Hadley, P., Nakanishi, T., Dekker, C., NOV 9 2001. Logic circuits with carbon nanotube transistors. *Science* 294 (5545), 1317–1320.

[Balzani et al. (2003)] Balzani, V., Credi, A., Venturi, M., 2003. Molecular logic circuits. *ChemPhysChem* 4 (1), 49–59.

[Belfield et al. (2002)]Belfield, K., Liu, Y., Negres, R., Fan, M., Pan, G., Hagan, D., Hernandez, F., SEP 2002. Two-photon photochromism of an organic material for holographic recording. *Chemistry of Materials* 14 (9), 3663–3667.

[Belousov (1959)] Belousov, B., 1959. A periodic reaction and its mechanism. *Compilation of Abstracts on Radiation Medicine*.

[Bennett (1982)] Bennett, C. H., Dec. 1982. The thermodynamics of computation - a review. *International Journal of Theoretical Physics* 21 (12), 905–940.

[Berlekamp et al. (1982)] Berlekamp, E., Conway, J., Guy, R., 1982. *Winning ways for your mathematical plays*. Academic Press.

[Blum (1998)] Blum, L., 1998. *Complexity and real computation*. Springer Verlag.

[Blum et al. (1988)] Blum, L., Shub, M., Smale, S., 1988. On a theory of computation over the real numbers; NP completeness, recursive functions and universal machines. In: Proceedings of the 29th Annual Symposium on Foundations of Computer Science. SFCS '88. IEEE Computer Society, Washington, DC, USA, pp. 387–397.

[Boolos et al. (2002)] Boolos, G., Burgess, J., Jeffrey, R., 2002. *Computability and logic*. Cambridge University Press.

[Brinkman et al. (1997)] Brinkman, W., Haggan, D., Troutman, W., dec 1997. A history of the invention of the transistor and where it will lead us. *Solid-State Circuits, IEEE Journal of* 32 (12), 1858 –1865.

[Bromley (1982)] Bromley, A. G., july-sept. 1982. Charles Babbage's analytical engine, 1838. *Annals of the History of Computing* 4 (3), 196 –217.

[Burch (2005)] Burch, C., 2005. Logisim: A graphical tool for designing and simulating logic circuits. http://ozark.hendrix.edu/~burch/logisim/ Retrieved Feb 2011

[Buss et al. (2001)] Buss, S. R., Kechris, A. S., Pillay, A., Shore, R. A., 2001. The prospects for mathematical logic in the twenty-first century. *The Bulletin of Symbolic Logic* 7 (2), pp. 169–196.

[Cardone and Hindley (2009)] Cardone, F., Hindley, J., 2009. Lambda-calculus and combinators in the 20th century. *Handbook of the History of Logic* 5, 723–817.

[Carol A. Jennings et al. (1997)] Carol A. Jennings, E. C., Marcel P. Breton, M. C., MaryAnna Isabella, M. C., Eric G. Johnson, P. C. F., Trevor I. Martin, B. C.,

John F. Oliver, C. C., 05 1997. Ink compositions with liposomes containing photochromic compounds. Patent, US 5633109.

[Chalfie et al. (1994)] Chalfie, M., Tu, Y., Euskirchen, G., Ward, W., Prasher, D., 1994. Green fluorescent protein as a marker for gene expression. *Science* 263 (5148), 802–805.

[Cheng et al. (2005)] Cheng, P.-N., Chiang, P.-T., Chiu, S.-H., 2005. A switchable macrocycle-clip complex that functions as a NOR logic gate. *Chem. Commun.*, 1285–1287.

[Church (1932)] Church, A., 1932. A set of postulates for the foundation of logic. *Annals of mathematics*.

[Church et al. (2006)] Church, A., Olszewski, A., Wolenski, J., Janusz, R., 2006. *Church's thesis after 70 years*. Vol. 1. Ontos Verlag.

[Church et al. (2012)] Church, G. M., Gao, Y., Kosuri, S., Aug. 2012. Next-Generation Digital Information Storage in DNA. *Science*.

[Collier et al. (1999)] Collier, C., Wong, E., Belohradsky, M., Raymo, F., Stoddart, J., Kuekes, P., Williams, R., Heath, J., JUL 16 1999. Electronically configurable molecular-based logic gates. *Science* 285 (5426), 391–394.

[Conrad (1985)] Conrad, M., May 1985. On design principles for a molecular computer. *Commun. ACM* 28, 464–480.

[Cook (2004)] Cook, M., 2004. Universality in elementary cellular automata. *Complex Systems* 15, 1–40.

[Cotogno (2003)] Cotogno, P., 2003. Hypercomputation and the physical Church-Turing thesis. *The British Journal for the Philosophy of Science* 54 (2), 181–223.

[Cronin et al. (2006)] Cronin, L., Krasnogor, N., Davis, B. G., Alexander, C., Robertson, N., Steinke, J., Schroeder, S., Khlobystov, A., Cooper, G., Gardner, P., Siepmann, P., Whitaker, B., 2006. The imitation game - a computational chemical approach to recognizing life. *Nature Biotechnology* 24, 1203–1206.

[de Silva et al. (1997)] de Silva, A. P., Gunaratne, H. Q. N., McCoy, C. P., 1997. Molecular photoionic and logic gates with bright fluorescence and "off-on" digital action. *Journal of the American Chemical Society* 119 (33), 7891–7892.

[de Silva and McClenaghan (2004)] de Silva, A. P., McClenaghan, N. D., 2004. Molecular-scale logic gates. *Chemistry: A European Journal* 10 (3), 574–586.

[DiCarlo et al. (2009)]DiCarlo, L., Chow, J. M., Gambetta, J. M., Bishop, L. S., Johnson, B. R., Schuster, D. I., Majer, J., Blais, A., Frunzio, L., Girvin, S. M., Schoelkopf, R. J., Jul. 2009. Demonstration of two-qubit algorithms with a superconducting quantum processor. *Nature* 460 (7252), 240–244.

[DiVencenzo (1995)] DiVencenzo, D. P., 1995. Quantum computation. *Science* 270 (5234), 255–261.

[Dürr and Bouas-Laurent (2003)] Dürr, H., Bouas-Laurent, H., 2003. *Photochromism: Molecules and Systems: Molecules and Systems*. Vol. 40. Elsevier Science.

[Dvornikov et al. (1994)] Dvornikov, A. S., Malkin, J., Rentzepis, P. M., 1994. Spectroscopy and kinetics of photochromic materials for 3d optical memory devices. *The Journal of Physical Chemistry* 98 (27), 6746–6752.

[Dvornikov et al. (2009)] Dvornikov, A. S., Walker, E. P., Rentzepis, P. M., DEC 10 2009. Two-Photon Three-Dimensional Optical Storage Memory. *Journal of Physical Chemistry A* 113 (49), 13633–13644.

[Elbaz et al. (2010)] Elbaz, J., Lioubashevski, O., Wang, F., Remacle, F., Levine, R. D., Willner, I., 2010. DNA computing circuits using libraries of DNAzyme subunits. *Nature Nanotechnology* 5, 417 – 422.

[Exelby and Grinter (1965)] Exelby, R., Grinter, R., 1965. Phototropy (or photochromism). *Chemical Reviews* 65 (2).

[Farsari et al. (1999)] Farsari, M., Huang, S., Birch, P., Claret-Tournier, F., Young, R., Budgett, D., Bradfield, C., Chatwin, C., Apr 1999. Microfabrication by use of a spatial light modulator in the ultraviolet: experimental results. *Opt. Lett.* 24 (8), 549–550.

[Felgner et al. (1987)] Felgner, P. L., Gadek, T. R., Holm, M., Roman, R., Chan, H. W., Wenz, M., Northrop, J. P., Ringold, G. M., Danielsen, M., 1987. Lipofection: a highly efficient, lipid-mediated DNA-transfection procedure. *Proceedings of the National Academy of Sciences* 84 (21), 7413–7417.

[Feringa et al. (2000)] Feringa, B., Delden, R. A., Koumura, N., Geertsema, E. M., 2000. Chiroptical molecular switches. *Chem. Rev* 100 (5), 1789–1816.

[Fischer and Hirshberg (1952)] Fischer, E., Hirshberg, Y., 1952. Formation of coloured forms of spirans by low-temperature irradiaton. *J. Chem. Soc*, 4522.

[Flood et al. (2004)] Flood, A. H., Stoddart, J. F., Steuerman, D. W., Heath, J. R., 2004. Whence molecular electronics? *Science* 306 (5704), 2055–2056.

[Forrester (1951)] Forrester, J., 1951. Digital information storage in three dimensions using magnetic cores. *Journal of Applied Physics* 22 (1), 44–48.

[Forster and Church (2006)] Forster, A. C., Church, G. M., Aug. 2006. Towards synthesis of a minimal cell. *Molecular Systems Biology* 2.

[Fredkin and Toffoli (2002)] Fredkin, E., Toffoli, T., 2002. *Collision-based computing*. Springer-Verlag, London, UK, Ch. Conservative logic, pp. 47–81.

[Frieder (1973)] Frieder, G., 1973. A balanced ternary computer. *Proceedings of the 3rd IEEE International Symposium on Multiple Valued Logic*, 68–75.

[Gardner (1970)] Gardner, M., 1970. Mathematical games: The fantastic combinations of John Conway's new solitaire game "life". *Scientific American* 223, 120–123.

[Gil et al. (2004)] Gil, R., Silva, F. J., Peretó, J., Moya, A., Sep. 2004. Determination of the core of a minimal bacterial gene set. *Microbiology and molecular biology reviews : MMBR* 68 (3), 518–537.

[Giordano et al. (2002)] Giordano, L., Jovin, T. M., Irie, M., Jares-Erijman, E. A., 2002. Diheteroarylethenes as thermally stable photoswitchable acceptors in photochromic fluorescence resonance energy transfer (pcFRET). *Journal of the American Chemical Society* 124 (25), 7481–7489.

[Goddard (1970)] Goddard, William A., L. J. J., March 1970. Direct access magnetic disc storage device, Patent 3503060.

[Görner and Matter (2001)] Görner, H., Matter, S., 2001. Photochromism of nitrospiropyrans: Effects of structure, solvent and temperature. *Physical Chemistry Chemical Physics* 3, 416–423.

[Green et al. (2007)] Green, J. E., Choi, J. W., Boukai, A., Bunimovich, Y., Johnston-Halperin, E., DeIonno, E., Luo, Y., Sheriff, B. A., Xu, K., Shin, Y. S., Tseng, H.-R., Stoddart, J. F., Heath, J. R., 2007. A 160-kilobit molecular electronic memory patterned at $10^{11}$ bits per square centimetre. *Nature* 445, 414–417.

[Greenfield (2006)] Greenfield, A., 2006. *Everyware: The Dawning Age of Ubiquitous Computing*. Berkeley, CA: New Riders.

[Guglielmetti (2003)] Guglielmetti, R., 2003. *Photochromism: Molecules and Systems*.

[Hájek (1998)]Hájek, P., 1998. *Metamathematics of fuzzy logic*. Vol. 4. Springer.

[Hammes (2002)] Hammes, G. G., 2002. Multiple conformational changes in enzyme catalysis. *Biochemistry* 41 (26), 8221–8228.

[Harris et al. (2011)] Harris, D. C., Saks, B. R., Jayawickramarajah, J., 2011. Protein-binding molecular switches via host-guest stabilized DNA hairpins. *Journal of the American Chemical Society* 133 (20), 7676–7679.

[Hay (1963)] Hay, L. S., 1963. Axiomatization of the infinite-valued predicate calculus. *The Journal of Symbolic Logic* 28 (1), pp. 77–86.

[Heinemann and Panke (2006)] Heinemann, M., Panke, S., Oct. 2006. Synthetic biology—putting engineering into biology. *Bioinformatics* 22 (22), 2790–2799.

[Hille (2001)] Hille, B., Jul. 2001. *Ion Channels of Excitable Membranes*, 3rd Edition. Sinauer Associates.

[Hirshberg (1956)] Hirshberg, Y., 1956. Reversible formation and eradication of colors by irradiation at low temperatures. a photochemical memory model. *J. Am. Chem. Soc* 78 (10).

[Ichino et al. (2008)] Ichino, T., Asahi, T., Kitahata, H., Magome, N., Agladze, K., Yoshikawa, K., 2008. Microfreight delivered by chemical waves. *The Journal of Physical Chemistry* 112 (8), 3032–3035.

[Inzelt (2012)] Inzelt, G., 2012. *Conducting polymers: a new era in electrochemistry*. Springer.

[Johnson et al. (2011)] Johnson, M. W., Amin, M. H. S., Gildert, S., Lanting, T., Hamze, F., Dickson, N., Harris, R., Berkley, A. J., Johansson, J., Bunyk, P., Chapple, E. M., Enderud, C., Hilton, J. P., Karimi, K., Ladizinsky, E., Ladizinsky, N., Oh, T., Perminov, I., Rich, C., Thom, M. C., Tolkacheva, E., Truncik, C. J. S., Uchaikin, S., Wang, J., Wilson, B., Rose, G., May 2011. Quantum annealing with manufactured spins. *Nature* 473 (7346), 194–198.

[Katz (2012)] Katz, E., 2012. *Molecular and Supramolecular Information Processing*. Wiley-VCH.

[Keyes (2001)] Keyes, R., mar 2001. Fundamental limits of silicon technology. *Proceedings of the IEEE* 89 (3), 227 –239.

[Kitano (2001)] Kitano, H., 2001. *Foundations of Systems Biology*. Mit Press.

[Kleene (1967)] Kleene, S., 1967. *Mathematical Logic*. John Wiley.

[Kleene (1952)] Kleene, S. C., Mar. 1952. *Introduction to Metamathematics*. Ishi Press International.

[Koçer et al. (2005)] Koçer, A., Walko, M., Meijberg, W., Feringa, B. L., 2005. A light-actuated nanovalve derived from a channel protein. *Science* 309 (5735), 755–758.

[Kolmakov et al. (2010)] Kolmakov, K., Belov, V., Bierwagen, J., Ringemann, C., Müller, V., Eggeling, C., Hell, S., 2010. Red-emitting rhodamine dyes for fluorescence microscopy and nanoscopy. *Chemistry - A European Journal* 16 (1), 158–166.

[Landauer (1961)] Landauer, R., 1961. Irreversibility and heat generation in the computing process. *IBM Journal of Research and Development* 5, 183–191.

[Le Naour-Sene (1981)] Le Naour-Sene, L., 09 1981. Process of integrating a photochromic substance into an ophthalmic lens and a photochromic lens of organic material. Patent, US 4286957.

[Lee et al. (2003)] Lee, J. N., Park, C., Whitesides, G. M., 2003. Solvent compatibility of poly(dimethylsiloxane)-based microfluidic devices. *Analytical Chemistry* 75 (23), 6544–6554.

[Lehn (1988)] Lehn, J.-M., 1988. Supramolecular chemistry-scope and perspectives molecules, supermolecules, and molecular devices (nobel lecture). *Angewandte Chemie International Edition in English* 27 (1), 89–112.

[Levenberg (1944)] Levenberg, K., 1944. A method for the solution of certain non-linear problems in least squares. *Quarterly Journal of Applied Mathmatics* II (2), 164–168.

[Levine and Davidson (2005)] Levine, M., Davidson, E. H., 2005. Gene regulatory networks for development. *Proceedings of the National Academy of Sciences of the United States of America* 102 (14), 4936–4942.

[Levitus et al. (1997)] Levitus, M., Talhavini, M., Negri, R. M., Atvars, T. D. Z., Aramendía, P. F., 1997. Novel kinetic model in amorphous polymers. spiropyran-merocyanine system revisited. *The Journal of Physical Chemistry B* 101 (39), 7680–7686.

[Li et al. (2009)] Li, H., Carter, J. D., LaBean, T. H., 2009. Nanofabrication by DNA self-assembly. *Materials Today* 12 (5), 24 – 32.

[Lin and Ma (2008)] Lin, J., Ma, D., 2008. Realization of write-once-read-many-times memory devices based on poly(N-vinylcarbazole) by thermally annealing. *Applied Physics Letters* 93 (9), 093505.

[Lin et al. (2010)] Lin, Y.-M., Dimitrakopoulos, C., Jenkins, K. A., Farmer, D. B., Chiu, H.-Y., Grill, A., Avouris, P., 2010. 100-GHz transistors from wafer-scale epitaxial graphene. *Science* 327 (5966), 662.

[Loss and DiVincenzo (1998)] Loss, D., DiVincenzo, D., Jan 1998. Quantum computation with quantum dots. *Physical Review A* 57 (1), 120–126.

[Lukasiewicz and Borkowski (1970)]Lukasiewicz, J., Borkowski, L., 1970. *Selected Works*. North-Holland Pub. Co.

[Malinowski (2007)] Malinowski, G., 2007. Many-valued logic and its philisophy. *The Many Valued and Nonmonotonic Turn in Logic* 8, 13.

[Mark (1999)] Mark, J., 1999. *Polymer Data Handbook*. Oxford University Press.

[Marriott et al. (2008)] Marriott, G., Mao, S., Sakata, T., Ran, J., Jackson, D. K., Petchprayoon, C., Gomez, T. J., Warp, E., Tulyathan, O., Aaron, H. L., Isacoff, E. Y., Yan, Y., 2008. Optical lock-in detection imaging microscopy for contrast-enhanced imaging in living cells. *Proceedings of the National Academy of Sciences* 105 (46), 17789–17794.

[McDonald's Corporation (2012)] McDonald's Corporation, August 2012. Chicken McNuggets Ingredient and Allergen Information. Retrieved September 2012. http://www.mcdonalds.co.uk/ukhome/product_nutrition.chicken.24.chicken-mcnuggets-6-pieces.html

[Mehta and Sen (2009)] Mehta, G., Sen, S., 2009. Towards a temperature-guided molecular switch: an unusual reversible low-temperature polymorphic phase transition in a conformationally locked environment. *Chem. Commun.*, 5981–5983.

[Menuel et al. (2008)] Menuel, S., Fontanay, S., Clarot, I., Duval, R. E., Diez, L., Marsura, A., 2008. Synthesis and complexation ability of a novel bis- (guanidinium)-tetrakis-(beta-cyclodextrin) dendrimeric tetrapod as a potential gene delivery (DNA and siRNA) system. study of cellular siRNA transfection. *Bioconjugate Chemistry* 19 (12), 2357–2362.

[Mitchison (1989)] Mitchison, T. J., 1989. Polewards microtubule flux in the mitotic spindle: evidence from photoactivation of fluorescence. *The Journal of Cell Biology* 109 (2), 637–652.

[Moore (1965)] Moore, G. E., 1965. Cramming more components onto integrated circuits. *Electronics Magazine* 4.

[Moya et al. (2009)] Moya, A., Krasnogor, N., Pereto, J., Latorre, A., 2009. Goethe's dream. challenges and opportunities for synthetic biology. *EMBO reports* 10, S1,, 28–32.

[Natarajan et al. (1992)] Natarajan, L. V., Tondiglia, V., Bunning, T. J., Crane, R. L., Adams, W. W., 1992. Liquid crystalline siloxanes containing spiropyran chromophores as reversible optical data storage materials. *Advanced Materials for Optics and Electronics* 1 (6), 293–297.

[Niemeyer (2002)] Niemeyer, C., 2002. The developments of semisynthetic DNA-protein conjugates. *Trends in Biotechnology* 20 (9), 395–401.

[Null and Lobur (2010)] Null, L., Lobur, J., 2010. *The essentials of computer organization and architecture*. Jones & Bartlett Learning.

[Ohya et al. (1998)] Ohya, Y., Okuyama, Y., Fukunaga, A., Ouchi, T., 1998. Photo-sensitive lipid membrane perturbation by a single chain lipid having terminal spiropyran group. *Supramolecular Science* 5 (1-2), 21 – 29.

[Pasparakis et al. (2009)] Pasparakis, G., Vamvakaki, M., Krasnogor, N., Alexander, C., 2009. Diol-boronic acid complexes integrated by responsive polymers - a route to chemical sensing and logic operations. *Soft Matter*.

[Paun (1998)] Paun, G., 1998. Computing with membranes. *Journal of Computer and System Sciences* 61, 108–143.

[Paun (2001)] Paun, G., 2001. P systems with active membranes: attacking np compete problems. *Automata, Languages and Combinatorics* 6 (1), 75–90.

[Păun (2006)] Păun, G., 2006. Introduction to membrane computing. *Applications of Membrane Computing*, 1–42.

[Perfilieva and Mockor (1999)] Perfilieva, I., Mockor, J., 1999. *Mathematical principles of fuzzy logic*. Vol. 517. Springer.

[Piccinini (2011)] Piccinini, G., 2011. The physical Church-Turing thesis: Modest or bold? *The British Journal for the Philosophy of Science*.

[Porcar et al. (2011)] Porcar, M., Danchin, A., de Lorenzo, V., dos Santos, V., Krasnogor, N., Rasmussen, S., Moya, A., 2011. The ten grand challenges of synthetic life. *Systems and Synthetic Biology* 5 (1-2), 1–9.

[Post (1936)] Post, E., 1936. Finite combinatory processes - formulation 1. *Journal of Symbolic Logic 1*, 103–105.

[Potgieter (2006)] Potgieter, P. H., 2006. Zeno machines and hypercomputation. *Theoretical Computer Science* 358, 23–33.

[Prasanna de Silva and McClenaghan (2000)] Prasanna de Silva, A., McClenaghan, N. D., 2000. Proof-of-principle of molecular-scale arithmetic. *Journal of the American Chemical Society* 122 (16), 3965–3966.

[Prasanna de Silva and Uchiyama (2007)] Prasanna de Silva, A., Uchiyama, S., 2007. Molecular logic and computing. *Nature Nanotechnology* 2, 399–410.

[Qi et al. (2012)] Qi, Z., Malo de Molina, P., Jiang, W., Wang, Q., Nowosinski, K., Schulz, A., Gradzielski, M., Schalley, C. A., 2012. Systems chemistry: logic gates based on the stimuli-responsive gel-sol transition of a crown ether-functionalized bis(urea) gelator. *Chem. Sci.* 3, 2073–2082.

[Radiation Protection Division, Health Protection Agency (2007)] Radiation Protection Division, Health Protection Agency, 2007. A non-binding guide to the artificial optical radiation directive 2006/25/EC.

[Radò (1962)] Radò, T., 1962. On non-computable functions. *Bell System Technical Journal* 41 (3), 877–884.

[Ramachandran et al. (2012)] Ramachandran, D., Liu, F., Urban, M. W., 2012. Self-repairable copolymers that change color. *RSC Adv.* 2, 135–143.

[Rao et al. (1996)] Rao, D., Aranda, F., Rao, D. N., Chen, Z., Akkara, J., Kaplan, D., Nakashima, M., 1996. All-optical logic gates with bacteriorhodopsin films. *Optics Communications* 127 (4-6), 193 – 199.

[Raymo and Giordani (2001)]Raymo, F. M., Giordani, S., 2001. Signal processing at the molecular level. *Journal of the American Chemical Society* 123 (19), 4651–4652.

[Raymo and Giordani (2002)]Raymo, F. M., Giordani, S., 2002. Multichannel digital transmission in an optical network of communicating molecules. *Journal of the American Chemical Society* 124 (9), 2004–2007.

[Reed et al. (1997)] Reed, M. A., Zhou, C., Muller, C. J., Burgin, T. P., Tour, J. M., 1997. Conductance of a molecular junction. *Science* 278 (5336), 252–254.

[Sakata et al. (2005)] Sakata, T., Yan, Y., Marriott, G., 2005. Optical switching of dipolar interactions on proteins. *Proceedings of the National Academy of Sciences of the United States of America* 102 (13), 4759–4764.

[Schenning et al. (2004)] Schenning, A., Jonkheijm, P., Hoeben, F., van Herrikhuyzen, J., Meskers, S., Meijer, E., Herz, L., Daniel, C., Silva, C., Phillips, R., Friend, R., Beljonne, D., Miura, A., Feyter, S. D., Zdanowska, M., Uji-i, H., Schryver, F. D., Chen, Z., Würthner, F., Mas-Torrent, M., den Boer, D., Durkut, M., Hadley, P., 2004. Towards supramolecular electronics. *Synthetic Metals* 147 (1-3), 43 – 48.

[Shannon (1938)] Shannon, C. E., 1938. *A symbolic analysis of relay and switching circuits*.

[Shinkai et al. (1981)] Shinkai, S., Nakaji, T., Ogawa, T., Shigematsu, K., Manabe, O., 1981. Photoresponsive crown ethers. 2. photocontrol of ion extraction and ion transport by a bis(crown ether) with a butterfly-like motion. *Journal of the American Chemical Society* 103 (1), 111–115.

[Shirakawa et al. (1977)] Shirakawa, H., Louis, E. J., MacDiarmid, A. G., Chiang, C. K., Heeger, A. J., 1977. Synthesis of electrically conducting organic polymers: halogen derivatives of polyacetylene, (CH). *J. Chem. Soc., Chem. Commun.*, 578–580.

[Shor (1994)] Shor, P. W., 1994. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. In: *Proceedings of the 35th Annual Symposium on Foundations of Computer Science*.

[Silva-Rocha and de Lorenzo (2008)]Silva-Rocha, R., de Lorenzo, V., 2008. Mining logic gates in prokaryotic transcriptional regulation networks. *FEBS Letters* 582 (8), 1237 – 1244.

[Singh (2011)] Singh, V., 2011. Lithography at 14nm and beyond: choices and challenges. In: *DAC'11*. pp. 459–459.

[Smaldon et al. (2010)] Smaldon, J., Romero-Campero, F. J., Trillo, F. F., Gheorghe, M., Alexander, C., Krasnogor, N., 2010. A computational study of liposome logic: towards cellular computing from the bottom up. *Systems and Synthetic Biology (Springer)* 4 (3), 157–179.

[Song et al. (2002)] Song, L., Jares-Erijman, E., Jovin, T., 2002. A photochromic acceptor as a reversible light-driven switch in fluorescence resonance energy transfer (FRET). *Journal of Photochemistry and Photobiology A: Chemistry* 150 (1-3), 177 – 185.

[Sony (2012)] Sony, September 2012. High-power blue-violet laser diode for blu-ray discs. Retrieved September 2012. http://www.sony.net/Products/SC-HP/-cx_news/vol40/pdf/sld3233vf.pdf

[Stankovich et al. (2006)] Stankovich, S., Dikin, D. A., Dommett, G. H. B., Kohlhaas, K. M., Zimney, E. J., Stach, E. A., Piner, R. D., Nguyen, S. T., Ruoff, R. S., Jul. 2006. Graphene-based composite materials. *Nature* 442 (7100), 282–286.

[Stitzel et al. (2006)] Stitzel, S., Byrne, R., Diamond, D., 2006. Led switching of spiropyran-doped polymer films. *Journal of Materials Science* 41, 5841–5844.

[Tamsir et al. (2011)] Tamsir, A., Tabor, J. J., Voigt, C. A., 2011. Robust multicellular computing using genetically encoded nor gates and chemical 'wires'. *Nature* 469, 212–215.

[Toffoli (1980)] Toffoli, T., 1980. Reversible computing. In: *Proceedings of the 7th Colloquium on Automata, Languages and Programming*. Springer-Verlag, pp. 632–644.

[Tominaga et al. (2001)] Tominaga, J., Mihalcea, C., Büchel, D., Fukuda, H., Nakano, T., Atoda, N., Fuji, H., Kikukawa, T., 2001. Local plasmon photonic transistor. *Applied Physics Letters* 78 (17), 2417–2419.

[Turing (1936)] Turing, A., 1936. On computable numbers, with an application to the entscheidungsproblem. *Proceedings of the London Mathematical Society*.

[Valeur (2001)] Valeur, B., Oct. 2001. *Molecular Fluorescence: Principles and Applications*. Wiley-VCH.

[Vitányi (2005)] Vitányi, P., 2005. Time, space, and energy in reversible computing. In: Proceedings of the 2nd conference on Computing frontiers. CF '05. ACM, New York, NY, USA, pp. 435–444.

[Vögtle et al. (1993)] Vögtle, F., Müller, W. M., Müller, U., Bauer, M., Rissanen, K., 1993. Photoswitchable catenanes. *Angewandte Chemie International Edition in English* 32 (9), 1295–1297.

[Volgraf et al. (2006)] Volgraf, M., Gorostiza, P., Numano, R., Kramer, R. H., Isacoff, E. Y., Trauner, D., Jan. 2006. Allosteric control of an ionotropic glutamate receptor with an optical switch. *Nat Chem Biol* 2 (1), 47–52.

[von Neumann (1945)] von Neumann, J., 1945. First draft of a report on the EDVAC, this incomplete report was distributed and incorrectly treated as published by many, resulting in some controversy over the attribution of the stored program concept to Von Neumann exclusively.

[von Neumann (1966)] von Neumann, J., 1966. *Theory of self-reproducing automata*. Urbana: University of Illinois Press.

[Wang (1960)]Wang, H., Apr. 1960. Proving theorems by pattern recognition I. *Commun. ACM* 3 (4), 220–234.

[Weber and Fussenegger (2011)] Weber, W., Fussenegger, M., 2011. Molecular diversity - the toolbox for synthetic gene switches and networks. *Current Opinion in Chemical Biology* 15 (3), 414 – 420.

[Winfree et al. (1998)] Winfree, E., Liu, F., Wenzler, L. A., Seeman, N. C., Aug. 1998. Design and self-assembly of two-dimensional DNA crystals. *Nature* 394 (6693), 539–544.

[Wohl and Kuciauskas (2005)] Wohl, C. J., Kuciauskas, D., 2005. Excited-state dynamics of spiropyran-derived merocyanine isomers. *J. Phys. Chem. B* 109, 22186–22191.

[Wojtyk et al. (2000)] Wojtyk, J. T. C., Wasey, A., Kazmaier, P. M., Hoz, S., Buncel, E., 2000. Thermal reversion mechanism of n-functionalized merocyanines to spiropyrans: A solvatochromic, solvatokinetic, and semiempirical study. *The Journal of Physical Chemistry A* 104 (39), 9046–9055.

[Wolfram (1983)] Wolfram, S., 1983. Statistical mechanics of cellular automata. *Reviews of Modern Physics* 55, 601–644.

[Wolfram (2002)] Wolfram, S., 2002. *A New Kind of Science*. Wolfram Media. http://www.wolframscience.com

[Yan et al. (2011)] Yan, Y., Marriott, M. E., Petchprayoon, C., Marriott, G., 2011. Optical switch probes and optical lock-in detection (OLID) imaging microscopy: high-contrast fluorescence imaging within living systems. *Biochemical Journal* 433 (3), 411–422.

[Zadeh (1965)] Zadeh, L., Jun. 1965. Fuzzy sets. *Information and Control* 8 (3), 338–353.

[Zhirnov et al. (2003)]Zhirnov, V., Cavin, R.K., I., Hutchby, J., Bourianoff, G., nov 2003. Limits to binary logic switch scaling - a gedanken model. *Proceedings of the IEEE* 91 (11), 1934 – 1939.