BioSystems 126 (2014) 12–26

Contents lists available at ScienceDirect

BioSystems

journal homepage: www.elsevier.com/locate/biosystems

Photochromic molecular implementations of universal computation

Jack C. Chaplin [a,b], Natalio Krasnogor [c,*], Noah A. Russell [a,**]

[a] Neurophotonics Lab, Schools of Biology, and Electrical and Electronic Engineering, University of Nottingham, Nottingham NG7 2RD, UK
[b] Institute for Advanced Manufacturing, Faculty of Engineering, University of Nottingham, Nottingham NG7 2RD UK
[c] Interdisciplinary Computing and Complex BioSystems (ICOS) Research Group, School of Computing Science, Newcastle University, Newcastle NE1 7RU, UK

ARTICLE INFO

Article history:
Received 2 June 2014
Received in revised form 11 August 2014
Accepted 3 September 2014
Available online 3 October 2014

Keywords:
Elementary cellular automata
Molecular switches
Photochromic molecules
Turing machines
Unconventional computing

ABSTRACT

Unconventional computing is an area of research in which novel materials and paradigms are utilised to implement computation. Previously we have demonstrated how registers, logic gates and logic circuits can be implemented, unconventionally, with a biocompatible molecular switch, NitroBIPS, embedded in a polymer matrix. NitroBIPS and related molecules have been shown elsewhere to be capable of modifying many biological processes in a manner that is dependent on its molecular form. Thus, one possible application of this type of unconventional computing is to embed computational processes into biological systems. Here we expand on our earlier proof-of-principle work and demonstrate that universal computation can be implemented using NitroBIPS. We have previously shown that spatially localised computational elements, including registers and logic gates, can be produced. We explain how parallel registers can be implemented, then demonstrate an application of parallel registers in the form of Turing machine tapes, and demonstrate both parallel registers and logic circuits in the form of elementary cellular automata. The Turing machines and elementary cellular automata utilise the same samples and same hardware to implement their registers, logic gates and logic circuits; and both represent examples of universal computing paradigms. This shows that homogenous photochromic computational devices can be dynamically repurposed without invasive reconfiguration. The result represents an important, necessary step towards demonstrating the general feasibility of interfacial computation embedded in biological systems or other unconventional materials and environments.

1. Introduction

Conventional computing involves the implementation of algorithmic processes to manipulate data on electronic hardware using binary logic. Unconventional computing on the other hand, uses new logical paradigms and new materials to build computational devices (Calude et al., 1998). Changing the physical materials used to compute opens up the possibility of embedding computers into biological systems at either a physiological or a cellular level. This could be achieved using biologically compatible molecular switches, for example. To demonstrate the feasibility of this possibility we have recently implemented registers and logic gates using photochromic molecular switches (Chaplin et al., 2012). Photochromic molecules (Exelby and Grinter, 1965) are a species of molecule with multiple stable forms. They can be reversibly switched between forms via the absorption of electromagnetic radiation. Spiropyrans are an example family of such photochromic molecules. (Berkovic et al., 2000). They possess a colourless leuco spiropyran form (SP) and a coloured trans-merocyanine form (MC). The transition of a sample of spiropyran molecules predominantly occupying the SP state to the MC state is called colouration, and the reverse is called decolouration.

One such spiropyran molecule is NitroBIPS (1′,3′-dihydro-1′,3′,3′-trimethyl-6-nitrospiro[2H-1-benzopyran-2-2′-2H-indole] or 6-nitro-BIPS or NBIPS). NitroBIPS is a spiropyran with a nitro group on the 6-position of the benzopyran section (Görner et al., 1996; Lenoble and Becker, 1986). The SP form absorption spectrum of NitroBIPS has its peak at ultraviolet wavelengths while the MC form absorption spectrum is in the visible range with a peak at green (Wohl and Kuciauskas, 2005), causing a solution predominantly in the MC state to appear purple or pink.

Absorption of green light by the MC form molecule generates the excited MC* fluorescent form, some molecules of which may immediately decay back to the MC form with the emission of a red photon (Görner, 1997), or undergo isomerization back to the SP

* Corresponding author: for matters of computer science. Tel.: +44 19 1208 5035.
** Corresponding author: for matters of engineering and photophysics. Tel.: +44 11 5846 8847.
E-mail addresses: jack.chaplin@nottingham.ac.uk (J.C. Chaplin), natalio.krasnogor@newcastle.ac.uk (N. Krasnogor), noah.russell@nottingham.ac.uk (N.A. Russell).

http://dx.doi.org/10.1016/j.biosystems.2014.09.004

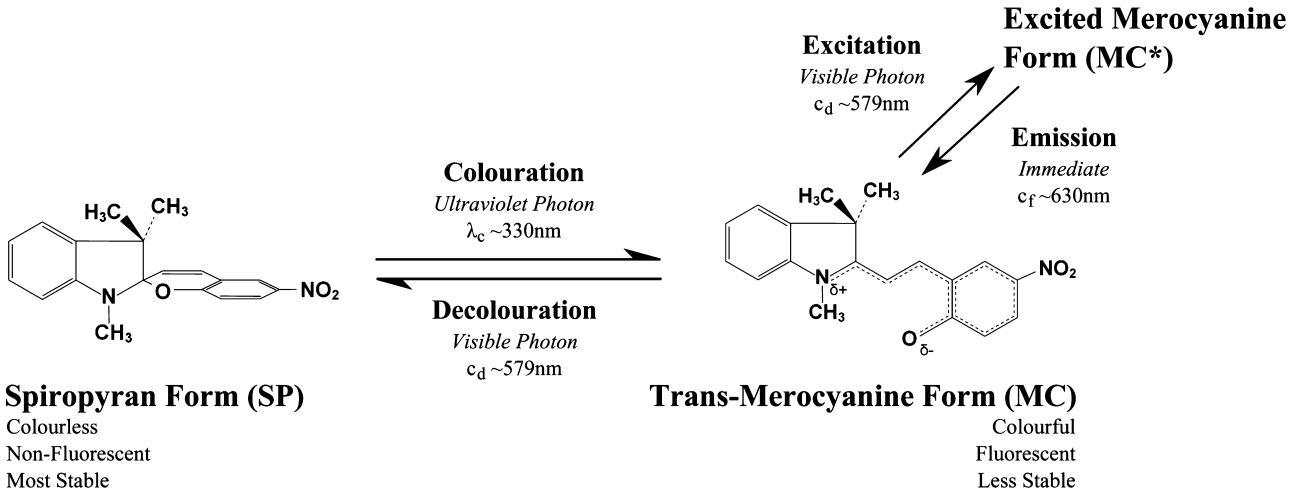


Fig. 1. The two stable forms of NitroBIPS and the transitions between them. As the spiropyran form is the most thermodynamically stable form, a population of NitroBIPS molecules will tend towards a majority-spiropyran equilibrium in the dark. A more detailed transition diagram can be found in Chaplin et al. (2012).

form. A small number of the excited MC* molecules will undergo irreversible bleaching. As fluorescence emission only occurs when the molecule is in the MC* form, the population of MC-form molecules can be estimated by the relative intensity of the emitted fluorescence signal when the sample is exposed to green light. These transitions are illustrated in Fig. 1. NitroBIPS is solvatochromic so the peak wavelength of absorption and the quantum yields (defined as the proportion of absorbed photons that cause a change in molecular form) for each transition are solvent dependent. Note that colouration and decolouration transitions also occur thermally. The SP to MC transition requires more energy than the reverse, so the majority of molecules are in the SP form at equilibrium at room temperature. The advantages of NitroBIPS (and other spiropyrans) compared to other species of photochromic molecules are that they possess higher quantum yields (Petchprayoon and Marriott, 2010), can be used with a wide variety of solvents (Görner and Matter, 2001) and are biologically compatible (Aizawa et al., 1977; Sakata et al., 2005b; Koçer et al., 2005; Ohya et al., 1998). However the MC form of NitroBIPS is subject to thermal relaxation (defined as the spontaneous, thermally driven reversion of a sample of molecules to an equilibrium state. For NitroBIPS the majority of molecules are in the SP form at equilibrium) at a moderate rate relative to other spiropyrans (Wojtyk et al., 2000).

We have previously shown that NitroBIPS can be used to implement registers and logic gates (Chaplin et al., 2012). Here we extend on this preliminary work and demonstrate an unconventional implementation of both parallel registers in the form of a Turing machine tape, as well as elementary cellular automata. Turing machine tapes and elementary cellular automata have been implemented as they both represent universal computing paradigms.

Alan Turing provided the first detailed theoretical description of a simple computational device that was capable of running any algorithm (Turing, 1936). Another simple universal computational device, which has been proven to be Turing complete (Cook, 2004), is the elementary cellular automaton. Elementary cellular automata consist of a one-dimensional array of 'cells' where the states of all cells are updated in parallel during each generation. Cells are updated according to their current state, the state of its two immediate neighbours and a predefined set of rules (Wolfram, 1983, 2002). Although Turing machines were introduced as mechanical models of mental processes, attempts have been made to mimic their structure via unconventional molecular processes (Qian et al., 2011; Rothemund, 1996; Shapiro and Karunaratne, 2001). Cellular automata, on the other hand, are typically used for modelling purposes (Nagel and Schreckenberg, 1992) but purpose specific hardware has been engineered to implement several versions of cellular automata (Shackleford et al., 2002; Gers et al., 1997).

The remainder of this paper is divided into five sections. Firstly we state the *Objectives* of this paper. Secondly, a *Methods* section which explains the production of NitroBIPS samples and details the illumination/detection hardware. The *Implementation* section, which builds on the theoretical work in (Chaplin et al., 2012), contains the theoretical basis for parallel registers, Turing machine tapes and cellular automata. Next, an *Experiments* section discusses the results of experiments carried out involving Turing machine tapes and elementary cellular automata. Lastly, a *Discussion* section discussing the strengths and weaknesses of this approach, and possible future directions.

2. Objectives

Our previous paper details the implementation of singular computational elements with photochromic molecules (Chaplin et al., 2012). The molecules were embedded in a polymer matrix and light pulses from LEDs were used to colourise and decolourise them. Emitted fluorescence was recorded with a photodiode. This allowed for data to be stored as the relative proportion of fluorescent molecules, and for logic gates and logic circuits to be executed by exploiting the floor and ceiling restrictions of the colourisation and decolourisation processes. The objective of the research in this paper was to expand upon this by designing and executing parallel computational elements, and implementing universal computation.

This paper builds multiple computational elements in parallel, by allowing a movable illumination area to address multiple regions in the polymer matrix. Parallel registers are implemented, and then expanded upon to implement the tape in a photochromic Turing machine. A combination of parallel registers and logic circuits are then used to implement elementary cellular automata, and hence a form of universal computation. Fig. 2 shows the relationship between these sections and previous work. The theoretical basis for each element is discussed first, followed by the experiments demonstrating these computational processes running on our experimental hardware.

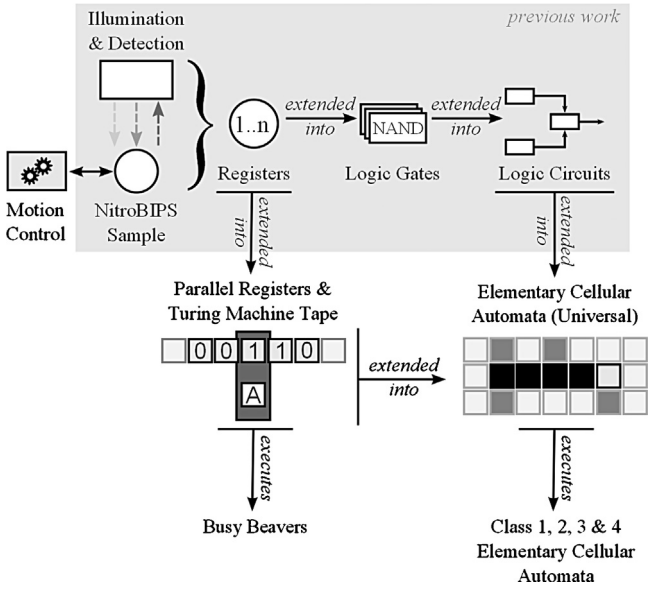


Fig. 2. An overview of this paper's aim. This paper draws and expands upon our previous paper (Chaplin et al., 2012), implementing new parallel functionality and a universal computing paradigm.

3. Methods

3.1. Samples

NitroBIPS in solution is difficult to work with over long periods of time because the solvents used tend to be volatile and evaporate. For our experiments, we require the volume and concentration to remain constant. We therefore encapsulated NitroBIPS in a polymer matrix to produce stable and easy to handle samples. NitroBIPS (273619 Aldrich) was first dissolved in methanol (a PDMS non-solvent) at a concentration of 4 mM and then mixed thoroughly at a ratio of 5:1 by volume with uncured polydimethylsiloxane (PDMS, Dow Corning Sylgard 184) containing the base and curing agent at a 10:1 ratio. The mixture was poured into a 90 mm diameter plastic Petri dish, to a height of 1.15 mm (weight 6900 mg) and left to degas and cure at room temperature for 48 h.

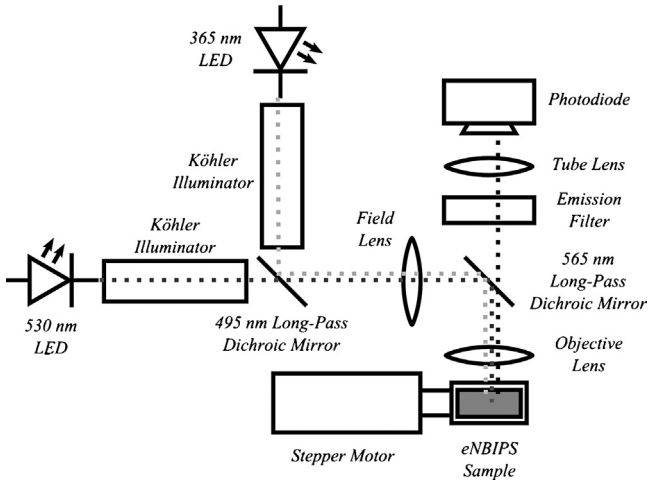


Fig. 3. Optical set-up used for the experiments in this paper. Uniform illumination from two light emitting diodes is produced using Köhler illuminators, combined with dichroic mirrors and imaged onto the sample. Fluorescence from the sample is detected by a photodiode. The region of the sample being illuminated can be controlled via the stepper motor, moving the sample relative to the illumination.

The cured samples of encapsulated NitroBIPS (eNBIPS) were then cut into pieces for use with the optical set-up.

3.2. Optics

An optical system, Fig. 3, was constructed to illuminate the eNBIPS samples using green (497–537 nm full width at half maximum bandwidth, 530 nm peak, Cairn Research) and UV (362–374 nm full width at half maximum bandwidth, 365 nm peak, Cairn Research) light emitting diode (LED) sources. An LED controller (OptoLED Power Supply, Cairn Research) was used to control the LED intensity and duration. A 507–543 nm bandpass filter was used to limit the green LED's emission spectrum. The green and UV beams were combined using a dichroic mirror (T495lpxr 495 nm long-pass filter, Chroma) and directed towards the sample plane using a second dichroic (565DCXR, 565 nm long-pass filter, Chroma). Köhler illuminators were constructed to create a uniform disc of illumination from each LED. The Köhler illuminators contain both aperture and field diaphragms. The aperture diaphragm was adjusted to minimise stray light in the optical path. A slit aperture (3.25 mm width) was placed in the plane of both field diaphragms. These slits were then imaged onto the eNBIPS sample, using a Nikon M-Plan 20× 0.35 numerical aperture objective lens, at an overall system demagnification of 8×, resulting in a 0.4 mm thick slit of illumination on the sample. Total illuminating power at the sample is 0.974 mW for green and 0.046 mW for UV. This rectangular slit aperture was used to achieve a high density of Turing machine and cellular automata cells; each with a larger area than a disc and therefore achieving a larger dynamic range from the sample. The eNBIPS sample was mounted to a Standa 8MS00-10-28 stepper motor, with a 10 mm travel range and a 0.156 μm 1/8th step size, which was controlled by a Standa 8SMC1-USBh stepper motor controller. The stepper motor allowed the sample to be translated so that the cell being illuminated could be rapidly changed. This movement could be achieved with a high degree of repeatability.

Fluorescence emission from the illuminated region of eNBIPS was then collected back through the objective lens, via the second dichroic, and imaged onto a photo-diode (FDS100, ThorLabs). A photo-diode amplifier (PDA200C, ThorLabs) converted current to voltage at a gain of 1×10^8 V/A. The signal was filtered (<20 Hz) using an 8-pole Kemo Benchmaster 8.41 filter and acquired using a National Instruments PCI-6221 data acquisition card. A custom LabVIEW program was written to acquire the fluorescence signal and to control the LEDs and the stepper motor.

4. Implementation

4.1. Photochromic parallel registers

Registers are fundamental computational components from which more advanced components are derived. The registers we implemented are described in terms of the proportion of molecular forms within a sample of NitroBIPS. As only the MC form is fluorescent, a higher proportion of MC-form molecules results in a higher level of measured fluorescence when the sample is exposed to excitatory (i.e. green) stimulus. As the proportion of MC-form molecules in the sample, the fluorescence emitted by the sample when exposed to excitatory stimulus, and the voltage measured by the photodiode in the system are all proportional we work with the measured voltages directly. The measured voltage from the photodiode spans the range from a minimally fluorescent state V_{MC*}^{min} to a maximally fluorescent state V_{MC*}^{max} (Fig. 4). The voltage can be incremented with colouration stimulus and decremented with decolouration stimulus between these limits. Each state of the register is defined by a corresponding

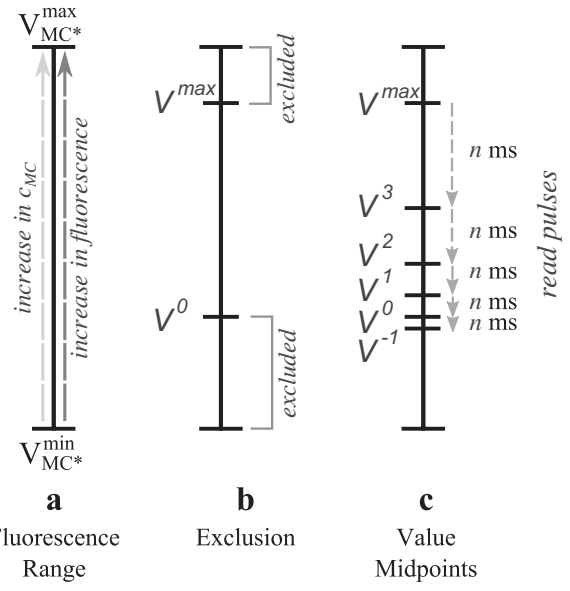


Fig. 4. Summary of register initialisation. The limits of the fluorescence range are determined, as shown in (a). The top and bottom of the range are excluded, as shown in (b) because colouration and decolouration are subject to diminishing returns as molecules switch form. By limiting the extremities of the range, we improve execution times at the expense of register range. The remaining fluorescence range is then divided into values (c), distributed such that a fixed duration read pulse always decolours by a single value. An additional −1 value is required for when V^0 is read.

level of fluorescence, as well as an upper and lower bound to allow for noise in the system.

As the fluorescence excitation stimulus wavelengths and the decolouration wavelengths are the same, using an excitation stimulus to read the register also decolours it. Two basic strategies for dealing with this issue were discussed in our previous paper (Chaplin et al., 2012). Here we use an alternative advanced method for parallel registers, which we refer to as decrement reading. This method requires the modification of the reading protocol such that a read pulse causes the sample to decolour by a single value, and so the register remains in a known state. The speed of colouration decreases exponentially as the sample colourises. Similarly, the speed of decolouration decreases exponentially as the sample decolours. As a result, the rate of decolouration is non-linear, and as the value of the register is not known in advance the separation of values must also be non-linear (Fig. 4).

As an additional refinement, we exclude the top and bottom regions of the fluorescence range as these regions are most affected by the non-linearity of colouration/decolouration rates (Fig. 4). By removing these regions, execution speed is improved at the cost of reduced register capacity. The excluded range can be varied, but we typically exclude the bottom 1/3 and top 1/6 of the fluorescent range (the larger exclusion at the bottom of the range is to account for an additional −1 value detailed later). As the trend in colouration/decolouration from $V^{1/3}_{MC*}$ to $V^{5/6}_{MC*}$ is approximately linear, transition times between values are similar, although not identical. To initialise a register, we first determine the rate of colouration with increasing duration UV pulses, followed by long green pulses that decolour to V^{min}_{MC*}. The peak intensities can then be fitted to the colouration and decolouration formulas that were discussed in detail in Chaplin et al. (2012). These formulas describe the recorded voltage at time t. For colouration:

$$V_{MC*}(t) = V^{max}_{MC*} - (V^{max}_{MC*} e^{-k_2 t}) + V_{offset} \quad (1)$$

where k_2 is a constant representing the rate of colouration. For decolouration:

$$V_{MC*}(t) = V^0_{MC*} e^{-k_1 t} + V_{offset} \quad (2)$$

where k_1 is a constant representing the rate of decolouration.

Secondly, we measured the noise in the fluorescence signal, determined by the standard deviation of the signal. The noise defines the dynamic range and imposes a limit on the capacity of the registers, i.e.

$$\text{DynamicRange} = \frac{V^{max}_{MC*} - V^{min}_{MC*}}{\text{noise}} \quad (3)$$

The register's dynamic range is then divided into N distinguishable values $\{V^0, V^1 \ldots V^{max}\}$, each separated by at least six standard deviations to give an error rate of <0.1% for Gaussian noise. These values were spaced over the range such that a fixed-duration decolouration pulse will decrement the value by exactly one. An additional value V^{-1} must also be calculated for cases where a register in V^0 is read. Values are also given upper and lower bounds, as noise and thermal relaxation will cause variation in fluorescence measurements. A summary of the initialisation process can be seen in Fig. 4.

In this way a value transition table can then be compiled containing illumination times for colouration and decolouration value transitions. For an example see Table 1. Decolouration by one value is equal to the read time, and decolouration by more than one value is a multiple of the read time. The table accounts for the decrementation in value of the register when reading it. This places the restriction, however, that a register can only be read once before having its value changed. Otherwise it could be decolourised below V^{-1} where illumination times are no longer defined. This transition table now allows the register to be incremented or decremented arbitrarily by exposing it to the appropriate duration of illumination.

By translating the area of illumination with the stepper motor, sub-regions of the sample are able to be addressed. As the NitroBIPS molecules are encapsulated in a polymer matrix, they are not subject to diffusion. This allows us to store multiple values in separate registers simultaneously along a single axis.

4.2. Photochromic Turing machine tape

The one-dimensional array of registers implemented as described above can be easily extended to Turing's model of a minimal computing device. A Turing machine is classified by the size of the alphabet, Γ, and the number of possible internal states (I-states), Q, such that an 'n-state m-symbol machine' (where n and m are integers) corresponds to $|Q| - 1$ (omitting the halting I-state) and $|\Gamma|$, respectively. The creation of a Turing machine using photochromic registers as the tape requires the implementation of the four fundamental components: the *tape*, *head*, *I-state register* and *transition function* as shown in Fig. 5. We have implemented these as follows:

1. *Tape*: A Turing machine tape is a bidirectional, limitless one-dimensional array of cells, with each cell storing a letter from an alphabet. A one dimensional sample of eNBIPS can store

Table 1
An example value transition table for a ternary register. The table is created by the initialisation process in our experiments. Times are given in milliseconds. Positive numbers are colouration times, negative numbers decolouration times. This table accounts for the decrementation of a value by the read action i.e. the illumination time from V_1 to V_0 is 0 as the read action already decremented the register by 1.

	To		
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

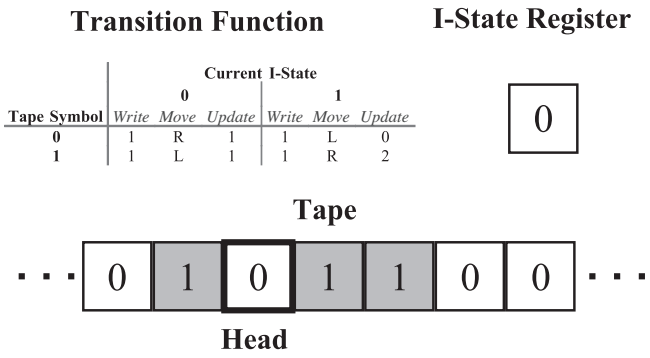


Fig. 5. Traditional Turing machine, with the four principle components. The tape is bidirectionally limitless and divided into cells, and the read/write head addresses a single cell at a time. The I-state register tracks the internal state of the Turing machine, and the transition function is the program for the machine, determining what actions to perform each step.

multiple values along its length. By subdividing the eNBIPS sample into non-overlapping cells, we can implement a register at each cell, and store a symbol from a defined alphabet Γ, provided $d \geq |\Gamma|$, where d is the capacity of the register. If $d < |\Gamma|$, k registers with base d could comprise a cell with capacity $d^k \geq |\Gamma|$. The tape is not unbounded, but has a finite length that is determined by the range of the stepper motor travel. In this early study, our illumination width is 0.4 mm and the motor travel is 10 mm, giving a total tape capacity of 25 cells. This limits the possible algorithms which can be executed, but is sufficient to demonstrate a proof-of-concept. We begin Turing machine execution in the centre of the tape, and terminate execution should the machine move outside the tape bounds.

2. *Head:* The head is the illuminating optical system. The tape is moved relative to the head using the stepper motor, which alters the illuminated cell.
3. *I-state register*: The internal state register stores one of a set of possible internal states, Q, including a start I-state and a halt I-state. Two options exist for implementation. Initially we stored the I-state on the controlling computer and updated it in response to the transition function *update* step. An alternative is to store the I-state in an eNBIPS register on the tape itself. In this case, we allocate the leftmost tape cell (or cells, if the register capacity is less than $|Q|$) as the I-state register and read/write to it during operation. An n-state m-symbol machine would require $\lfloor \log_m n \rfloor + 1$ I-state cells. If it is stored on tape, the Turing machine I-state is stored at the leftmost end. For simplicity we retain the same base as the rest of the Turing machine so that the I-state cells do not need to be initialised separately during the *initialise* action.
4. *Transition function*: A transition function, δ, is a two-dimensional table defining which operations to perform for each combination of *observed* cell value and current I-state. It consists of 3-tuples of instructions, to *print, move* and *update*. We store the transition function on the controlling computer but it could also be stored on the tape.

In addition to the Turing machine components, the atomic actions *observe, decide, print, move* and *update*, as well as an *initialise* and an *output* action, must be implemented to work with the four components.

1. *Initialise*: The system must be initialised before use. This involves initialising the registers as described above. As the eNBIPS sample is homogenous, the initialisation process only has to be run once and the values calculated for one register are used for all registers. Additionally, all registers must be set to the Turing machine blank symbol or to a specified input sequence.
2. *Observe*: Reading the symbol on the tape cell currently addressed by the head is equivalent to reading an eNBIPS register. The read pulse decolourises the register by one value, so a new symbol is always written during the subsequent *print* action.
3. *Decide*: The *decide* action involves cross-referencing the *observed* symbol and the I-state with δ to determine the next course of action. If the I-state is stored on tape, this requires moving to and reading the leftmost register on the tape. Again, though reading will decrement the I-state register(s), a new I-state is always written during the *update* action (if the I-state is stored on the tape). The result of the *decide* action is the three-tuple of actions (*print, move* and *update*) taken from the transition function.
4. *Print*: Writing a symbol to a tape cell is the same as writing a value to a register. A colouration or decolouration stimulus, with a duration determined by the value transition table is used to set the register. The current value of the register is known from the preceding *observe* action, albeit decremented by one as a consequence of decrement reading.
5. *Move*: A *move* operation involves moving the head one cell left or right on the tape by incrementing or decrementing the stepper motor position by the cell width to address the next parallel register. An attempt to move beyond the bounds of the tape terminates execution with an '*exceeded tape bound*' error. This action is also extended to allow for movement to/from the I-state register(s).
6. *Update*: This action updates the I-state in accordance with the transition function. This requires changing a value in the controlling computer, or moving to the leftmost cell(s) and writing to the I-state register. Altering the I-state is the same as writing to any other cell on the tape; the value is known from the *decide* action, and the stimulation required is given by the value transition table.
7. *Output*: The output of a Turing machine is given by the list of symbols stored on the tape after the halt I-state has been reached. Following a halt I-state, the *output* action reads every cell on the tape (including the I-state if relevant) and outputs the read value to the user (on screen or as a text file).

4.3. Photochromic elementary cellular automata

Cellular automata are discrete models often used in the modelling of physical systems. Elementary cellular automata are the simplest form of cellular automata. An elementary cellular automaton is a bidirectional one-dimensional array, C, of cells, c_i, that each store an element from a binary set of states. The initial conditions define the state each cell begins in. Every cell is updated simultaneously once per iteration according to a transition function, F which is dependent on the state of the cell and its two immediate neighbours, c_{i-1} and c_{i+1}, i.e. $\forall i, c'_i = F(c_{i-1}, c_i, c_{i+1})$. New iterations are calculated until a predefined generation limit, g_{max}, is reached. Elementary cellular automata have no other halting condition. As an example, the famous universal Rule 110 (Cook, 2004) has the transition function shown in Fig. 6. Rule 110 is the only elementary cellular automaton that has been proven to be universal (apart from trivial reflections and inversions of Rule 110). The universality of elementary cellular automata makes them one of the simplest universal computing paradigms, alongside Turing machines.

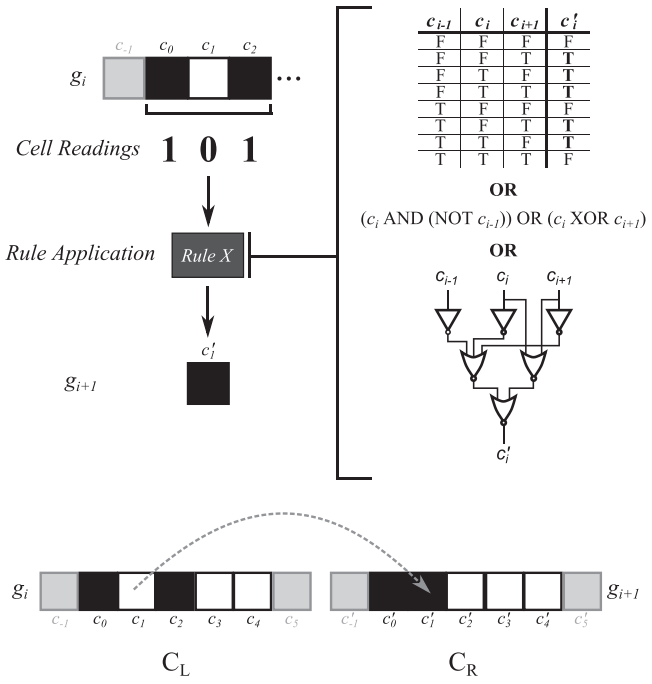


Fig. 6. General execution of an elementary cellular automaton. For each cell, the cell and its two immediate neighbours are read, their values entered into the transition function, and the cell updated with the new value for the subsequent generation. The transition function can be represented as a truth table, a logic function or a logic circuit. The transition functions shown here are for Rule 110. The result from this function is then written to the corresponding cell in a duplicate array. In this example, transition functions are being calculated from C_L and written sequentially to C_R. Once a generation is complete, the origin array (C_L here) will be cleared, and the results of rule applications to cells in C_R written to C_L.

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 to be held at 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 cells are automatically added as the pattern grows.

The implementation of elementary cellular automata with our system bears many resemblances to the implementation of Turing machines. It requires the implementation of three key components; the *right/left arrays*, the *head* and the *transition function*:

1. *Right/left arrays:* The cellular array *C* is implemented in the same manner as the tape in the Turing machine. In the current hardware implementation, the *head* cannot update all of the cells simultaneously. The results of each cell update must therefore be stored in a separate array. Therefore we define a pair of duplicate arrays; a *right* and a *left array* (C_R and C_L). They are implemented on a uniform eNBIPS sample, with parallel registers addressed by the read/write *head*. Each register stores a binary digit representing the value of the cell; 0 or 1.To improve execution speed, the *update* process is altered to first copy the results from C_L into C_R, then from C_R back to C_L. For both arrays, the number of cells is finite as the eNBIPS sample is itself finite. For a sample *S* with a maximum capacity of |*S*|, the arrays C_R and C_L are of size $\lfloor |S|/2 \rfloor$.For finite samples a boundary condition must be imposed; we chose that cells external to the sample boundaries are always value 0.
2. *Head*: As mentioned, in the current hardware configuration the *head* cannot address every element in the array simultaneously, so instead updates one at a time. The *head* is the optical system, and moves the sample relative to the optics to address specific cells.
3. *Transition function*: The transition function, *F* can be expressed in several forms as shown in Fig. 6. However, as logic circuits can be executed in eNBIPS samples (Chaplin et al., 2012), we have chosen to implement the transition functions as a three-input logic circuit where the circuit is executed in c_i and determines the value of c'_i.

Implementation of elementary cellular automata also requires four atomic actions in a manner similar to a Turing machine; *move*, *observe*, *execute* and *print*:

1. Move: Though *move* is not present in traditional elementary cellular automata definitions, the system requires the *head* to move in order to address cells sequentially. Unlike the Turing machine *move*, elementary cellular automata *moves* will not necessarily move one cell at a time, as it must move between C_R and C_L to write c'_i.
2. Observe: The *observe* action involves reading the symbol written to the tape cell the *head* is currently addressing. Like Turing machine *observe* actions, this is equivalent to reading a register. The state of the cell must be reset after the *observe* action, as future cells will need to *observe* its value during their *execute* action. As reading a cell causes it to be decremented by one, *observe* actions also require the cell be incremented by one after the read pulse.
3. Execute: To determine the next state of the cell, a logic circuit that represents the transition function is executed, with the states of c_{i-1}, c_i and c_{i+1} as inputs. The output of the logic circuit is the new value for the cell.
4. Print: When the new value for c_i is determined, *print* applies colouration or decolouration stimulus to the cell to set the value appropriately.

5. Experiments

5.1. Photochromic Turing machine

In order to demonstrate the Turing machine operation, we first implemented busy beaver algorithms. Busy beavers (Radò, 1962) are Turing machines with a state transition table that achieves the maximum amount of work before halting when run on an unbounded tape with all cells set to blank. Two measurements of work are typically used. These are: the generalized busy beaver function, $\Sigma(n, m)$, which is the number of non-blank symbols on the tape at the end of execution of an *n*-state, *m*-symbol machine; and the generalized maximum shifts function, $S(n, m)$, where the work is defined by the number of shifts (i.e. *move* actions) made by the *n*-state, *m*-symbol machine. For each class of *n*-state, *m*-symbol Turing machine, there is at least one champion machine for each definition of work. The champion is the Turing machine which is believed to be the machine for which Σ or *S* is the highest, and yet which halts. Note that the busy beaver problem has been shown to be equivalent to the halting problem, and hence is uncomputable (Radò, 1962). As the busy beaver functions are uncomputable, the only way to prove a champion is to enumerate all busy beavers of that class. We chose busy beavers to test the system as they are the smallest machines that produce the maximally long output, requiring a complex series of *move* and *print* actions, while still halting. The tape is initialised as described for parallel registers.

a) Transition Function

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

b) Execution

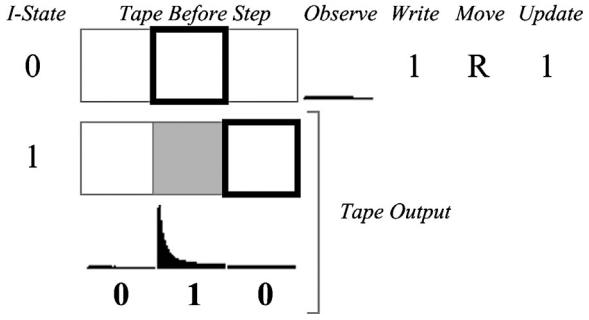


Fig. 7. A 1-state 2-symbol busy beaver champion for Σ and S. This machine has values $\Sigma = 1$ and $S = 1$. (a) is the transition function for this particular champion (there are several tied champion machines) in table form, and (b) shows the execution of the Turing machine as executed on our experimental system. Each row is a step in the Turing machine's execution, with the I-state (in this case stored on the controlling computer), the tape and the position of the head (bold), the fluorescence recorded during the observe action, and the actions determined by the transition function. The tape output section shows the state of the tape at the end of execution, the fluorescence measured during the output action and the symbols this corresponds to.

However, as the duration for which a value must be stored in a Turing machine is not known in advance and varies, Turing tapes are susceptible to thermal relaxation. To alleviate this, the difference between V^0 and V^1 is increased, and long colouration times used (60 s), to increase the time a register in V^1 takes to thermally relax to V^0.

Examples of busy beavers as executed on our system can be seen in Figs. 7–9, representing 1-state 2-symbol, 2-state 2-symbol, and 3-state 2-symbol busy beavers. Each figure shows *a*; the transition function for that Turing machine and *b*; the execution of that Turing machine on our optical system with eNBIPS samples prepared as described earlier. The left-most column shows the value of the I-state and (for the latter two figures), the recorded fluorescence from the I-state cells saved on-tape. The centre grid shows the tape for each iteration, with the position of the head (bold). As all the implemented Turing machines are two-symbol machines, the cells are either V^0 (white) or V^1 (grey). Note that the cell states are implicit as not all cells are read every iteration. To the right of the tape are four columns, representing the fluorescence waveform recorded when the current cell was *observed*, the symbol that should be *written* to the cell (V^x, in accordance with the transition function), the direction of *movement* of the head, and the new I-state to be *updated*. The bottom section of each figure shows the result of the *output* action; the cell values and recorded waveforms.

Note that it is also possible to store the transition function on the tape, although we did not do this because the number of cells on our tape is limited. As the transition function is static, any reads of the transition function would need to be followed by a new *reset* action to undo the *read* decrement. Though previously

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

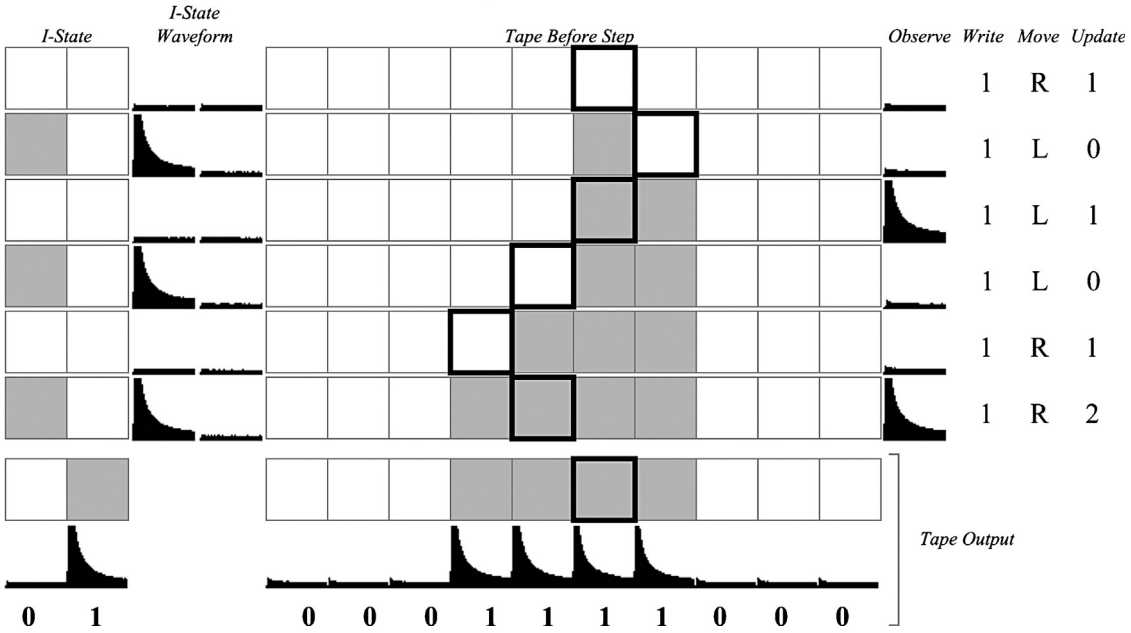


Fig. 8. The 2-state 2-symbol busy beaver champion for both Σ and S. This machine has values $\Sigma = 4$ and $S = 6$. (a) is the transition function for machine in table form, and (b) shows the execution of the Turing machine as executed on our experimental system as described in Fig. 7, but now with the I-state stored on tape.

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

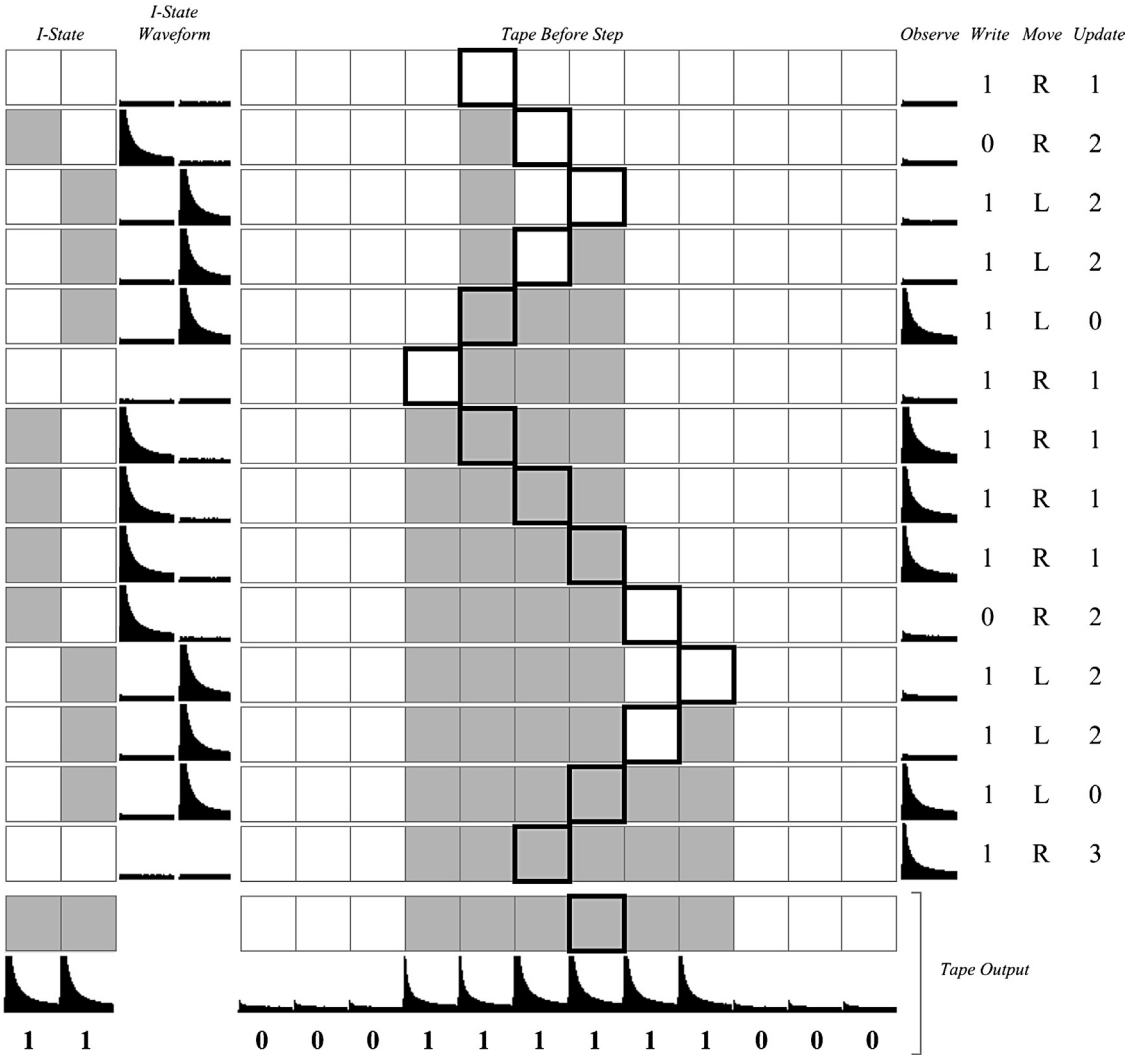


Fig. 9. The 3-state 2-symbol busy beaver champion for Σ (but not S). This machine has values $\Sigma = 6$ and $S = 14$. (a) is the transition function for machine in table form, and (b) shows the execution of the Turing machine as described in Fig. 7.

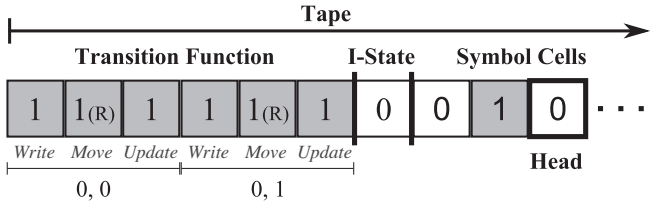
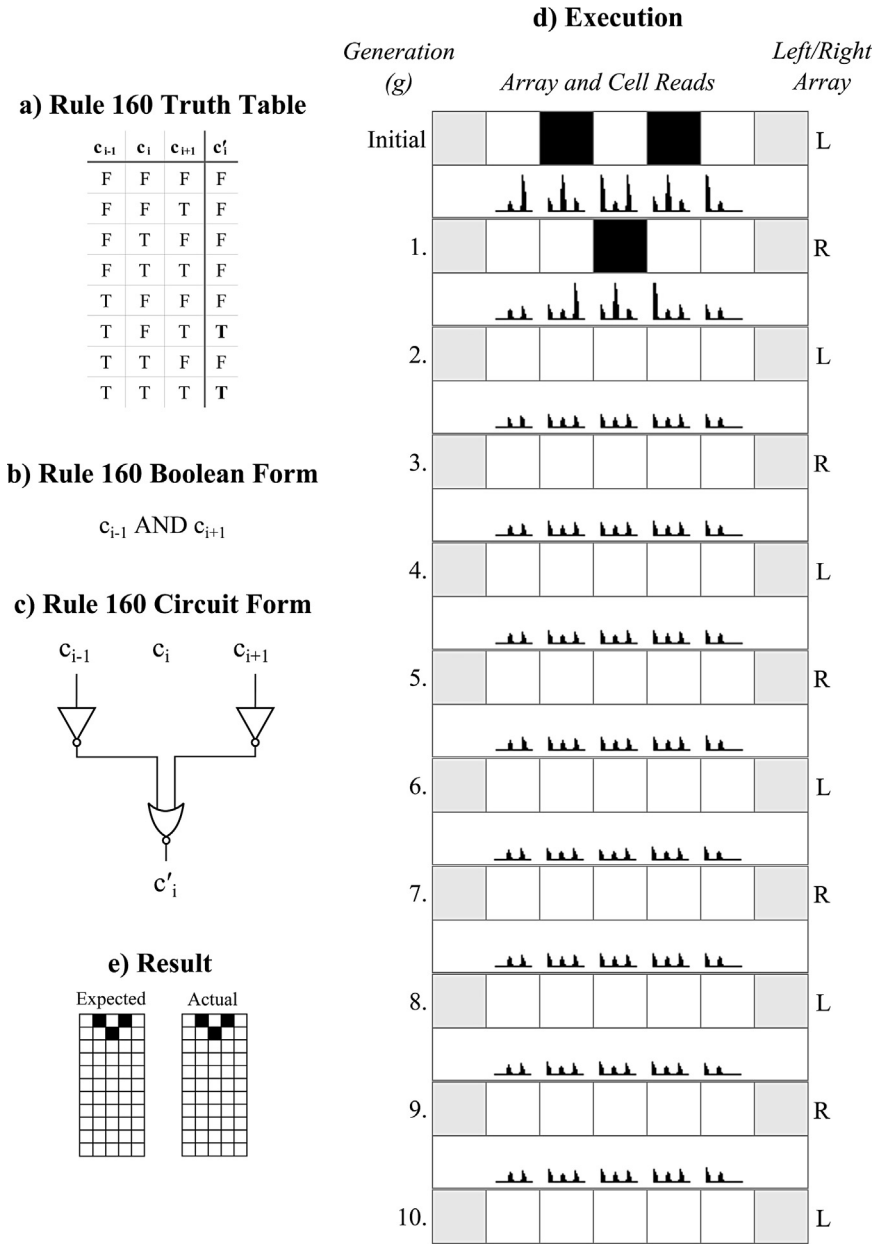


Fig. 10. One possible representation of a transition function stored on tape. This approach has the transition function, I-state and Turing tape all on the same encapsulated NitroBIPS sample. Two virtual boundaries (bold lines) subdivide the tape into three. The tuples are unlabeled in this example – their position is implicit – but tuples could also have labels at the cost of additional space. The L/R symbols are represented as 0 (left) and 1 (right).

we have rendered transition functions in the form of a table for readability, it only needs to be expressed as a set of tuples. For example, the 1-state, 2 symbol busy beaver transition function in Fig. 7 could be rendered as 2 tuples: $\{(0, 0 \rightarrow 1, R, 1), (0, 1 \rightarrow 1, R, 1)\}$. These tuples could then be transcribed to the tape, as shown in Fig. 10.

5.2. Photochromic elementary cellular automata

We have implemented at least one elementary cellular automaton from each Wolfram Class (Wolfram, 2002). Wolfram



c_{i-1}	c_i	c_{i+1}	c'_i
F	F	F	F
F	F	T	F
F	T	F	F
F	T	T	F
T	F	F	F
T	F	T	**T**
T	T	F	F
T	T	T	**T**

Fig. 11. Execution of Rule 160. Rule 160 is a class 1 elementary cellular automaton, and rapidly converges to a uniform state of 'off' cells. The initial condition was two 'on' cells. (a, b and c) show three versions of the transition function, as a table, logic function and logic circuit respectively. (d) shows the execution of the elementary cellular automata as executed on our system. Generation shows the current generation (limit 10). The centre shows the current array and boundary virtual cells, the symbols written to it (white = 0, grey = 0 virtual cell, black = 1), and the fluorescence measurements recorded when each cell and its neighbours are read. Left/Right array shows which array is currently in use for that generation (see Fig. 6). e shows the expected and experimentally recorded results of the experiment.

classes describe the complexity of the behaviour of an elementary cellular automata rule. The four classes are defined as:

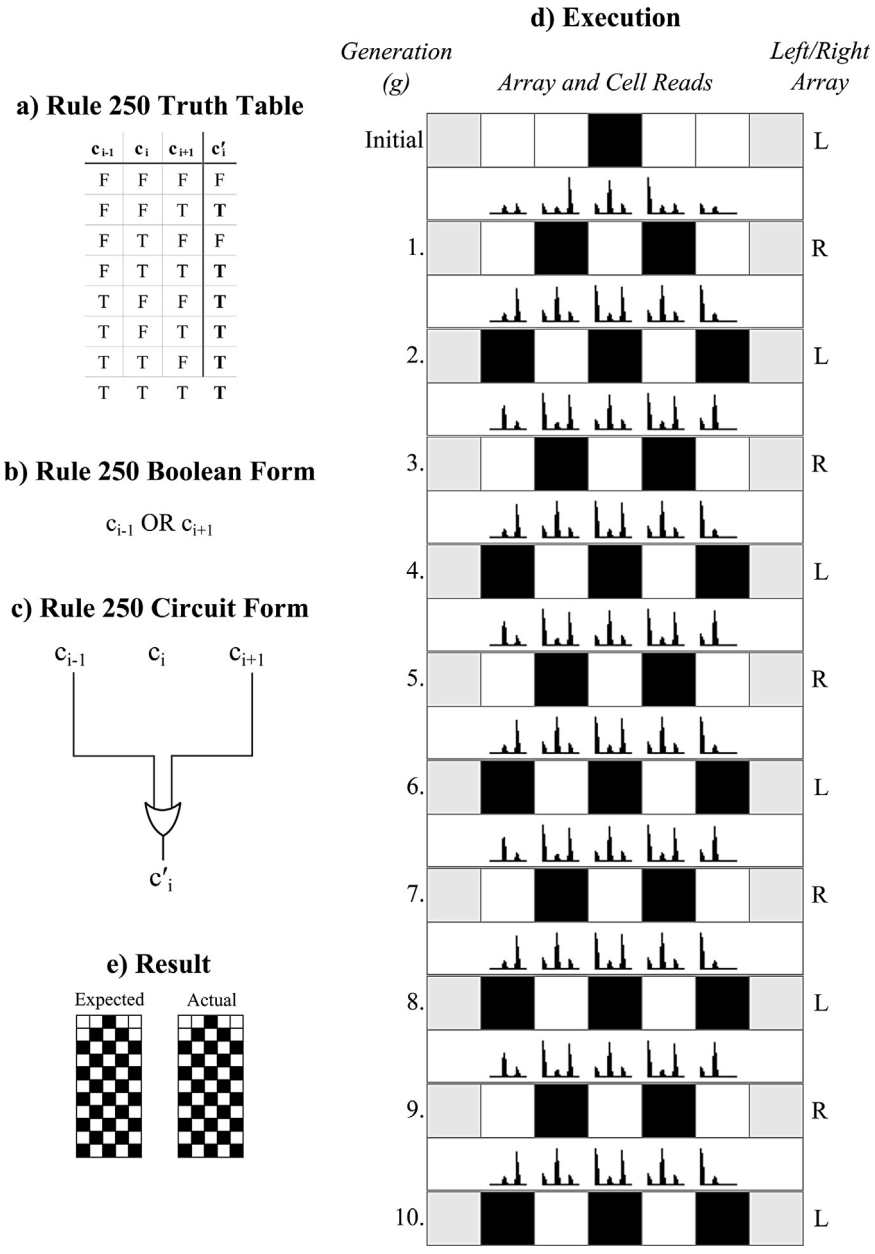
Class 1: Almost all initial states evolve into a uniform final state or alternates between two states.

Class 2: Many uniform or repetitive final states exist, but they exist of repeating simple structures.

Class 3: The elementary cellular automaton evolves in an apparently random or unpredictable manner.

Class 4: Localised structures form and interact in 'interesting' ways.

Their executions are shown below in Figs. 11–15. Each figure has five components: *a, b* and *c* are three representations of the transition function, as a table, Boolean function and logic circuit respectively. *d* shows the execution of the elementary cellular automata, with ten generations with an array size of 5 and with two additional 'virtual' cells beyond the array boundary, fixed at value 0. Each generation consists of two rows. The first is the graphical representation of the cell states, either 0 (white), 1 (black) or a virtual cell (grey). The second consists of the three waveforms recorded for each cell's three necessary *observe* actions,



c_{i-1}	c_i	c_{i+1}	c'_i
F	F	F	F
F	F	T	**T**
F	T	F	F
F	T	T	**T**
T	F	F	**T**
T	F	T	**T**
T	T	F	**T**
T	T	T	**T**

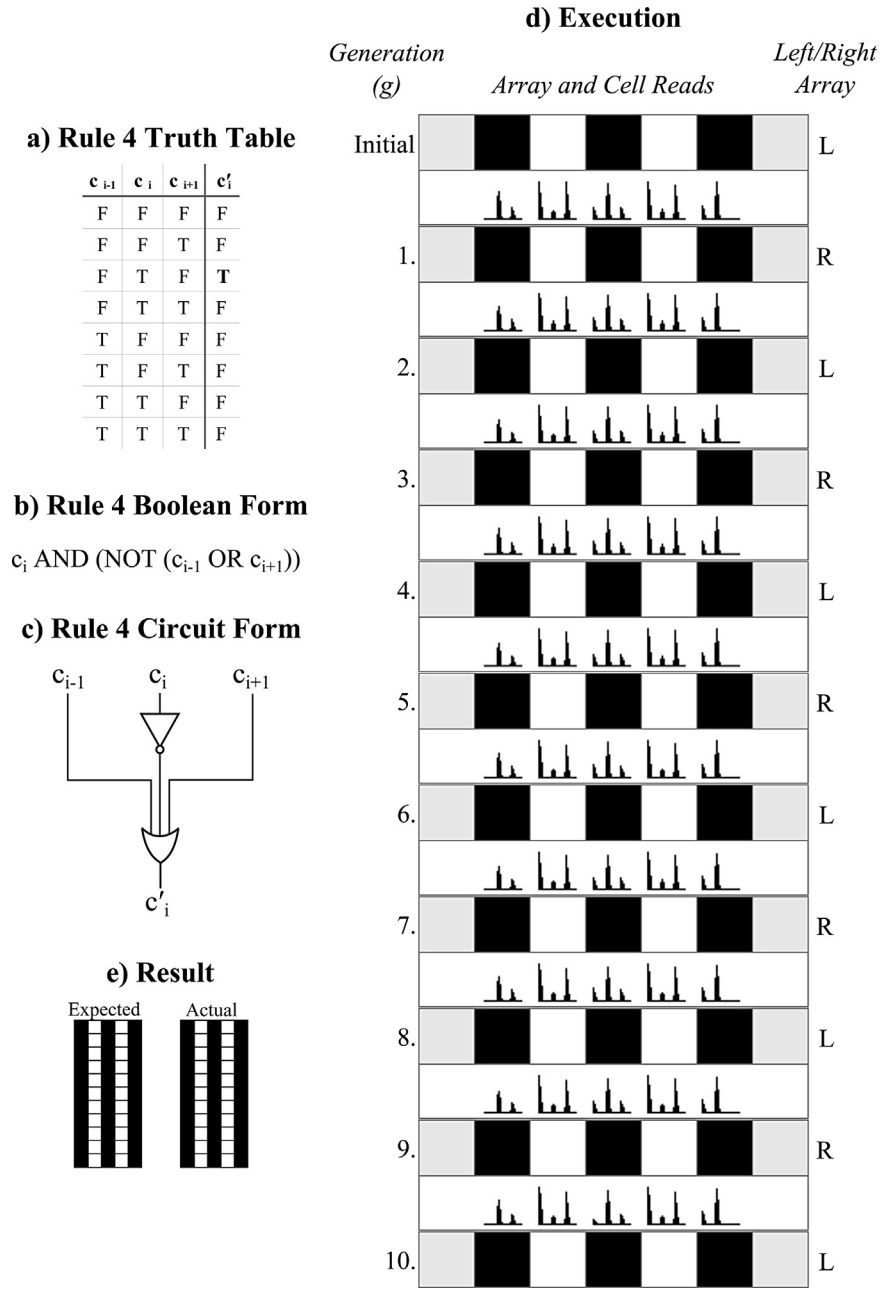
Fig. 12. Execution of Rule 250. Rule 250 is a class 1 elementary cellular automaton, though the fixed values of the boundary cells causes it to form a repeating alternating pattern. This is a limitation of the chosen boundary condition. The initial condition was a single on cell. See Fig. 11 for a detailed description.

showing the fluorescence from c_{i-1}, c_i and c_{i+1}. To the left is the generation number, and to the right is which array is being read from. *e* is a comparison of the result given by execution, and the expected result.

Unlike Turing machine tapes, the time between reads of a cell in an elementary cellular automaton is fixed, as each cell is observed and written each generation. However, the time required to execute the transition function increases with the complexity of the circuit. For the most complex rule presented here – Rule 110 – 27 min are required per generation. The array size of five for experiments here is chosen as it's the largest array size that can be executed with Rule 110, without a cell thermally relaxing to an incorrect value. For example, this effect can be seen in Fig. 15; 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. This is a limitation of the current hardware set-up, and is not intrinsic to the method.

6. Discussion

In this paper we have implemented photochromic molecular-based parallel registers, used these to implement tapes in Turing machines, and extended these techniques to implement elementary cellular automata. In combination with the illumination and



c_{i-1}	c_i	c_{i+1}	c'_i
F	F	F	F
F	F	T	F
F	T	F	**T**
F	T	T	F
T	F	F	F
T	F	T	F
T	T	F	F
T	T	T	F

Fig. 13. Execution of Rule 4. Rule 4 is a class 2 elementary cellular automaton, forming repetitive structures. The initial condition is alternating cell states. See Fig. 11 for a detailed description.

measurement system, these both provide examples of universal computing paradigms (Turing, 1936). Parallel registers and Turing machine tapes demonstrate the ability for eNBIPS samples to store multiple values in sub-regions, and elementary cellular automata build upon this by demonstrating how logic circuits can be executed at the same site as registers, without sample alteration. These techniques could be applied to other spiropyrans or modified to work with other species of molecular switches.

There are several limitations with the hardware in its current form, but these issues are surmountable. Colouration/decolouration times are long due to low light power at the sample, and as a result registers may thermally relax. More powerful light sources are available however. For example, cheap solid state lasers with at least two orders of magnitude more power are readily available. The efficiency of the optics could also be optimised. Smaller illumination areas, two dimensional addressing of regions and parallel illumination of multiple areas could be easily achieved using a spatial light modulator and a camera to measure the emitted fluorescence. Using this approach it would be possible to implement parallel reconfigurable arrays of logic gates, circuits and registers all in the same hardware. Three dimensional addressing is also possible with two-photon excitation. (Parthenopoulos and Rentzepis, 1989), although this would have to be performed serially.

As demonstrated in this paper, it is possible to implement universal computation using a combination of NitroBIPS and

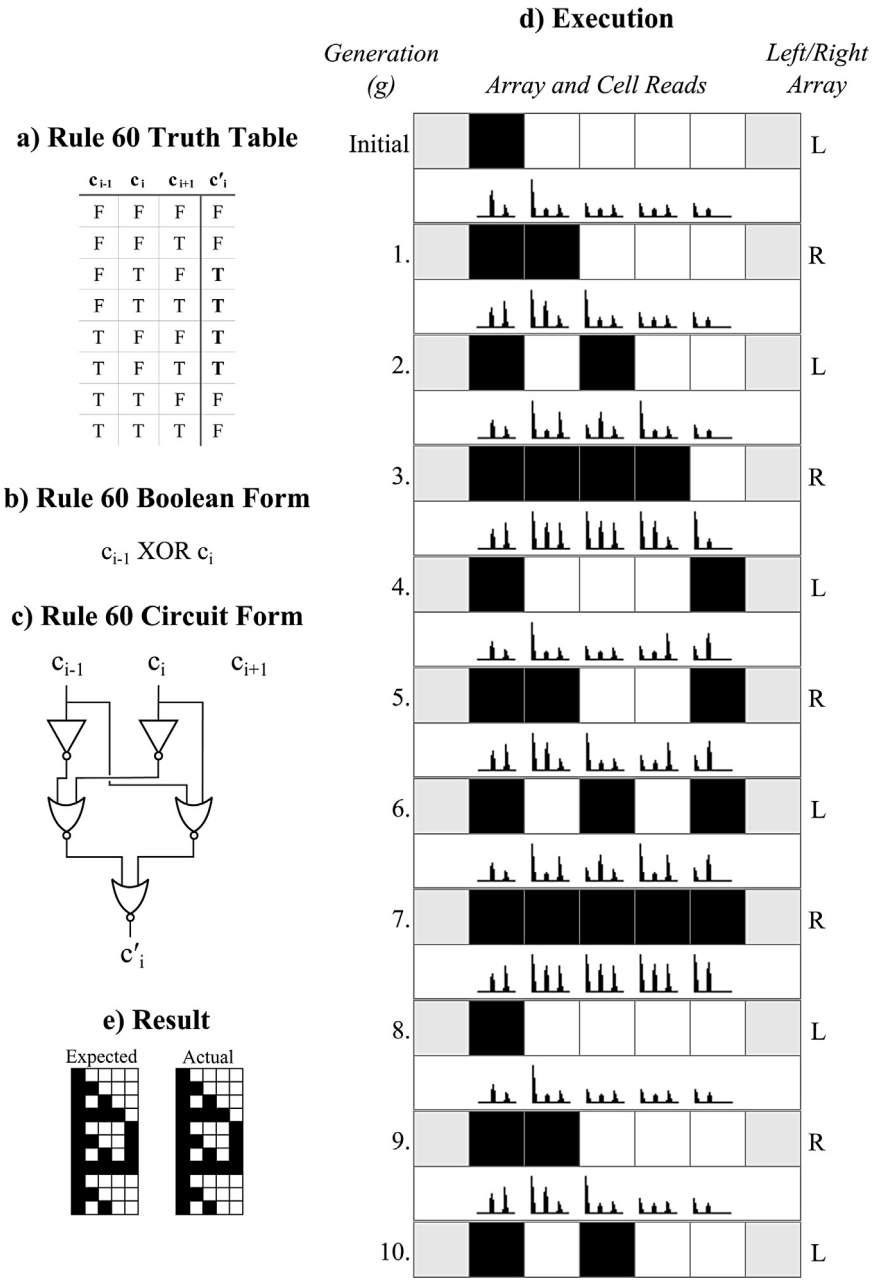
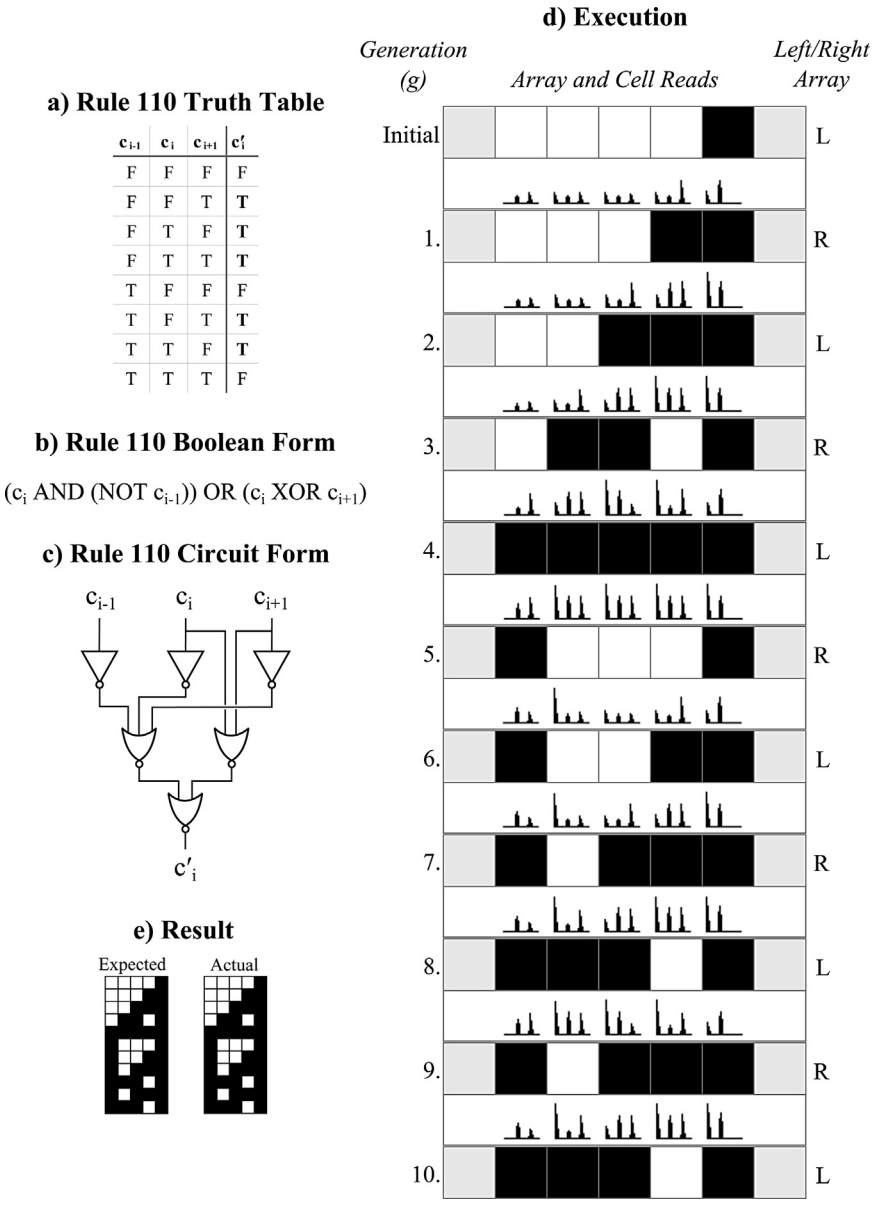


Fig. 14. Execution of Rule 60. Rule 60 is a class 3 elementary cellular automaton, forming random structures. The initial condition is $c_0 = 1$. See Fig. 11 for a detailed description.

external control hardware. By allowing the state of registers (either independently, or as Turing machine/elementary cellular automata cells) within a computational device to affect, or be affected by biological processes, the biological system can become embedded in the computation. In principle embedded bio-molecular computers could allow a more sophisticated exchange of information between external systems and biological systems. This could be useful for the monitoring and control of both healthy and diseased cells and physiological systems. Photochromic components could also offer input/output methods to synthetic biological chells (Cronin et al., 2006), where the general purpose and reusable nature of photochromic logic could allow for on-demand spatially targeted alteration of chell behaviour or reporting of chell status.

Photochromic molecular switches show promise for biologically compatible embedded computational devices because they can be observed and controlled remotely using light. Compared to other potential molecular computing methods such as DNA computing or enzymes, spiropyrans are small molecules that can be coupled to a multitude of biological processes and the form of these molecules can be used to affect those processes. For example, proteins can have their binding properties altered (Sakata et al., 2005b); membrane permeability can be perturbed to allow molecules to cross (Ohya et al., 1998); and DNA sequences can be blocked from transcription (Andersson et al., 2008). Many other possibilities exist, both with NitroBIPS, other photochromic molecules and other molecular switches.

As an example, a NitroBIPS molecule tethered to a protein is subject to different dipolar forces depending on the form of the molecule [12]. These dipolar interactions cause a change in the NitroBIPS molecule's orientation relative to the protein as it



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

Fig. 15. Execution of Rule 110. Rule 110 is a class 4 elementary cellular automaton, and is proven to be capable of emulating a Turing machine, and hence of universal computation (Cook, 2004). The initial condition is $c_5 = 1$. See Fig. 11 for a detailed description.

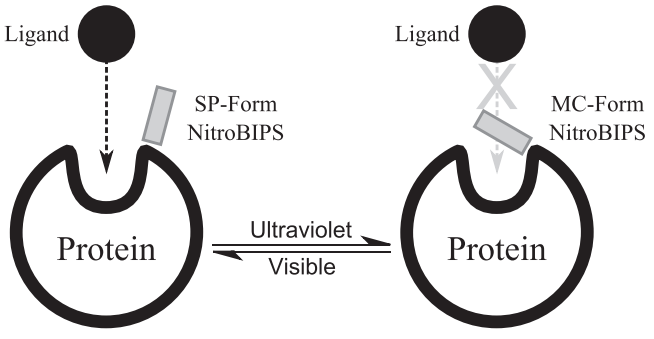


Fig. 16. NitroBIPS can be bound to proteins. A molecule in the MC form inhibits protein binding more than a SP form molecule (Sakata et al., 2005a). Hence, in a population of NitroBIPS-tagged proteins, a higher concentration of merocyanine-form molecules will cause an overall reduction in the rate of protein binding. The population could be addressed as a photochromic register, incrementing or decrementing the value with light and the value of the register determines the rate of protein binding.

changes from SP to MC form; with the MC form having a greater dipole moment. If this change in orientation were near a ligand binding site on the protein, the association constant of ligand binding could be reduced in the MC form, but not the SP form, allowing for the reversible inhibition of protein functions (Sakata et al., 2005a; Specht et al., 2009), as in Fig. 16. Alternatively, if the quantum yields of NitroBIPS transitions can be biased due to changing dipolar interactions resulting from protein binding or structural reconfigurations then biological processes could directly influence computations via changes to register values or logic gate operation.

As an additional example, a small unilamellar vesicle (SUV) composed of single chain lipids can have a spiropyran introduced at the hydrophobic tail (Ohya et al., 1998). Spiropyrans bound in this way preserve the hydrophilic–hydrophobic–hydrophilic bi-layer formation of an SUV when in the SP form. However, upon irradiation with UV light, the spiropyrans change form to the MC form. MC-form molecules bound to lipids are hydrophilic and

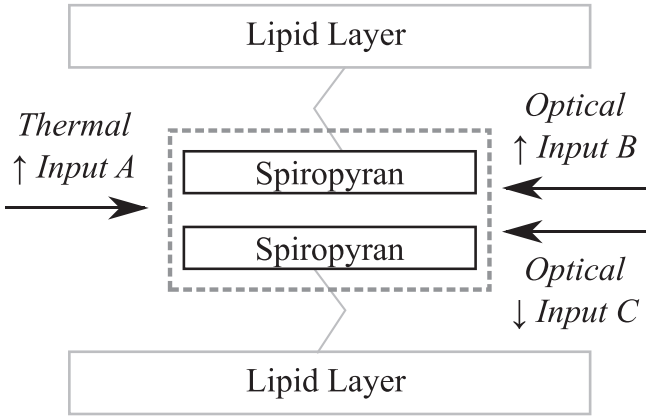
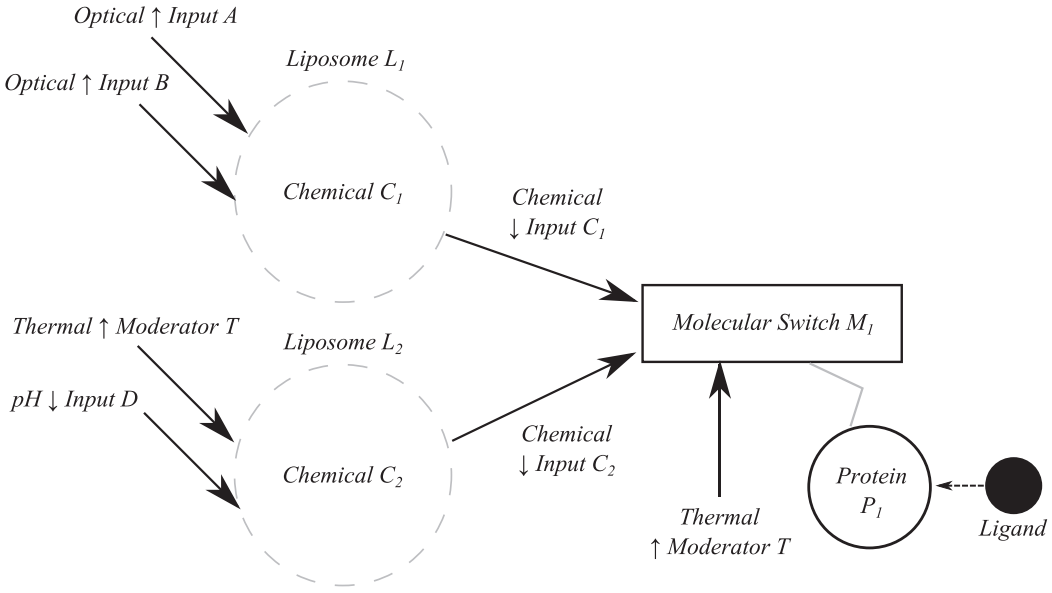


Fig. 17. A hypothetical spiropyran forms a liposome logic gate. This gate has a thermal input and two optical inputs, one each for colouration and decolouration, forming a three-input logic gate (A OR B) AND (NOT C). Raising the temperature or applying a colouration stimulus causes the release of the contents of the liposome (representing a True output). However, both processes would be inhibited by the presence of decolouration stimulus (False output).

disrupt the bi-layer. This perturbation causes the release of the contents of the SUV into the surrounding environment, but crucially does not cause the complete breakdown of the liposome. The release of internal contents can hence be controlled using optical stimulation to increase or decrease the permeability of the membrane. This mechanism could be utilised in two ways; in a manner similar to photochromic registers, where the 'value' now corresponds to permeability of the liposome; or in a manner similar to logic gates, where a combination of input stimuli determine the permeability. This latter option is particularly useful if the molecular switch can be coupled to a biological signalling pathway and act as an input to the gate; see Fig. 17 for an example. Both mechanisms could allow for the targeted release of drugs e.g. chemotherapeutic agents where the drugs are cytotoxic. Targeted drug delivery using liposomes has already been shown (Gabizon et al., 2004), but the combination of liposomal logic (Smaldon et al., 2009, 2010) and external influence via photochromic logic gates and registers could offer more complex targeting and condition reactive options.

All examples shown here require external illumination and fluorescence detection; however as the form of the spiropyran



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
$C_1 \parallel C_2$: P_1 binding permitted.
$\neg C_1$ & $\neg C_2$: P_1 binding inhibited.

A	*B*	*D*	C_1	C_2	P_1 Binding
F	F	F	F	T	**F**
F	F	T	F	F	**T**
F	T	F	F	T	**F**
F	T	T	F	F	**T**
T	F	F	F	T	**F**
T	F	T	F	F	**T**
T	T	F	T	T	**F**
T	T	T	T	F	**F**

Fig. 18. The creation of a biologically-compatible logic circuit with a complex combination of molecular switches bound to liposomes and proteins. Hypothetical molecular switches, sensitive to light (in L_1), pH (in L_2) and chemicals (M_1) are used to control the binding of protein P_1 and a ligand. The temperature of the system also affects the circuit, causing positive thermal moderation (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 to the ligand by blocking the active site. If either C_1 or C_2 are present, M_1 folds and the ligand can bind. This represents a (A AND B) NOR (NOT D) circuit.

molecule bound to an internal cellular process can alter that process, some reduction in reliance on external hardware could be achieved. Despite this, it is unlikely that photochromic computation could operate independent of external control, nor as a substitute for general-purpose computing *ex vivo* due to the slow speed of colouration/decolouration reactions and the noise in fluorescence readings. It is more likely that photochromic computation would be used as an input/output method to interface with biological processes or other molecular implementations of logic gates, allowing for the creation of complex biologically compatible logic circuits e.g. Fig. 18.

In summary, we have demonstrated that universal computation can be implemented, using photochromic molecular switches, in the form of Turing machines and elementary cellular automata. Furthermore these photochromic computational devices can be dynamically repurposed without invasive reconfiguration and have great potential for embedding within biological systems.

Acknowledgements

This work was supported by the Engineering and Physical Sciences Research Council, grant numbers EP/E018580/1, EP/H022112/1, EP/H024905/1, and EP/J004111/1; and by the Biotechnology and Biological Sciences Research Council, grant number BB/F01855X/1.

References

Aizawa, M., Namba, K., Suzuki, S., 1977. Photo control of enzyme activity of [alpha]-amylase. Arch. Biochem. Biophys. 180 (1), 41–48.
Andersson, J., Li, S., Lincoln, P., Andréasson, J., 2008. Photoswitched DNA-binding of a photochromic spiropyran. J. Am. Chem. Soc. 130 (36), 11836–11837.
Berkovic, G., Krongauz, V., Weiss, V., 2000. Spiropyrans and spirooxazines for memories and switches. Chem. Rev. 100 (5), 1741–1754.
Calude, C.S., Casti, J.L., Dinneen, M.J. (Eds.), 1998. Unconventional Models of Computation. 1st ed. Springer-Verlag Singapore Pte. Limited, New York.
Chaplin, J., Russell, N., Krasnogor, N., 2012. Implementing conventional logic unconventionally: photochromic molecular populations as registers and logic gates. Biosystems 109, 35–51.
Cook, M., 2004. Universality in elementary cellular automata. Complex Syst. 15, 1–40.
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. Nat. Biotechnol. 24, 1203–1206.
Exelby, R., Grinter, R., 1965. Phototropy (or photochromism). Chem. Rev. 65 (2) .
Görner, H., Matter, S., 2001. Photochromism of nitrospiropyrans: effects of structure, solvent and temperature. Phys. Chem. Chem. Phys. 3, 416–423.
Görner, H., Atabekyan, L.S., Chibisov, A.K., 1996. Photoprocesses in spiropyran-derived merocyanines: singlet versus triplet pathway. Chem. Phys. Lett. 260 (1–2), 59–64.
Görner, H., 1997. Photoprocesses in spiropyrans and their merocyanine isomers: effects of temperature and viscosity. Chem. Phys. 222, 315–329.
Gabizon, A., Shmeeda, H., Horowitz, A.T., Zalipsky, S., 2004. Tumor cell targeting of liposome-entrapped drugs with phospholipid–anchored folic acid–PEG conjugates. Adv. Drug Deliv. Rev. 56 (8), 1177–1192.
Gers, F., Garis, H.D., Korkin, M., 1997. Codi-1bit: a simplified cellular automata based neuron model. Lect. Notes Comput. Sci. 1363, 315–333.
Koçer, A., Walko, M., Meijberg, W., Feringa, B.L., 2005. A light-actuated nanovalve derived from a channel protein. Science 309 (5735), 755–758.
Lenoble, C., Becker, R.S., 1986. Photophysics, photochemistry, kinetics, and mechanism of the photochromism of 6′-nitroindolinospiropyran. J. Phys. Chem. 90 (1), 62–65.
Nagel, K., Schreckenberg, M., 1992. A cellular automaton model for freeway traffic. J. Phys. I 2 (12), 2221–2229.
Ohya, Y., Okuyama, Y., Fukunaga, A., Ouchi, T., 1998. Photo-sensitive lipid membrane perturbation by a single chain lipid having terminal spiropyran group. Supramol. Sci. 5 (1–2), 21–29.
Parthenopoulos, D.A., Rentzepis, P.M., 1989. Three-dimensional optical storage memory. Science 245 (4920), 843–845.
Petchprayoon, C., Marriott, G., 2010. Preparation, characterization, and application of optical switch probes. Curr. Protoc. Chem. Biol. 2 (3), 153–169.
Qian, L., Soloveichik, D., Winfree, E., 2011. Efficient Turing-universal computation with DNA polymers. DNA Computing and Molecular Programming. Vol. 6518 of Lecture Notes in Computer Science. Springer, Berlin/Heidelberg, pp. 123–140.
Radò, T., 1962. On non-computable functions. Bell Syst. Tech. J. 41 (3), 877–884.
Rothemund, P.W.K., 1996. Method for construction universal dna based molecular turing machine. US Patent US5843661 A.
Sakata, T., Yan, Y., Marriott, G., 2005a. Family of site-selective molecular optical switches. J. Org. Chem. 70 (6), 2009–2013.
Sakata, T., Yan, Y., Marriott, G., 2005b. Optical switching of dipolar interactions on proteins. Proc. Natl. Acad. Sci. U. S. A. 102 (13), 4759–4764.
Shackleford, B., Tanaka, M., Carter, R.J., Snider, G., 2002. High-performance cellular automata random number generators for embedded probabilistic computing systems. In: Proceedings of the 2002 NASA/DoD Conference on Evolvable Hardware (EH'02). pp. 191–200.
Shapiro, E., Karunaratne, K.S.G., 2001. Method and system of computing similar to a Turing machine. US Patent US6266569 B1.
Smaldon, J., Krasnogor, N., Alexander, C., Gheorghe, M., 2009. Liposome logic. Proceedings of the 11th Annual conference on Genetic and evolutionary computation. GECCO '09. ACM, New York, NY, USA, pp. 161–168.
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. Syst. Synth. Biol. (Springer) 4 (3), 157–179.
Specht, A., Bolze, F., Omran, Z., Nicoud, J.-F., Goeldner, M., 2009. Photochemical tools to study dynamic biological processes. HFSP J. 3 (4), 255–264.
Turing, A., 1936. On computable numbers, with an application to the entscheidungsproblem. Proc. Lond. Math. Soc. 2 (42), 230–265.
Wohl, C.J., Kuciauskas, D., 2005. Excited-state dynamics of spiropyran-derived merocyanine isomers. J. Phys. Chem. B 109, 22186–22191.
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. J. Phys. Chem. A 104 (39), 9046–9055.
Wolfram, S., 1983. Statistical mechanics of cellular automata. Rev. Mod. Phys. 55, 601–644.
Wolfram, S., 2002. A New Kind of Science. Wolfram Media, Champaign, IL.