Towards Standalone In-Materio Devices: Stable Logic Gates and Elementary Cellular Automata in Carbon Nanotubes Material

Sigve Sebastian Farstad and Stefano Nichele and Gunnar Tufte
Norwegian University of Science and Technology
Trondheim, Norway
Email: {sigveseb@stud, nichele@idi, gunnart@idi}.ntnu.no

***Abstract*—Evolution-in-Materio (EIM) is a method of using artificial evolution to exploit physical properties of materials for computation. It has previously been successfully used to evolve a multitude of different computational devices implemented in physical materials. One of the main problems has been the stability of solutions due to physical changes in the substrate materials, e.g. topological or electrical, leading to devices sensitive to the evolutionary history, i.e. instability. This paper presents stable linearly and non-linearly separable logic gates, and elementary cellular automata (CA) transition functions, successfully evolved in single-walled carbon nanotube (SWCNT) and polymer composite materials. The results herein show the feasibility of building stand-alone in-materio devices and outline a novel computational abstraction for EIM based on CA. This work is done within the European Project NASCENCE.**

I. Introduction

The natural evolutionary process, in stark contrast to the traditional human top-down building-block-oriented engineering approach, is not one of abstraction, componentization and careful manipulation based on understanding of underlying mechanics. Rather, it is a ”blind-yet-guided” metaheuristic approach able to exploit properties to create ”designs” without needing to understand them or the underlying mechanics that power them. Evolution-in-Materio [1] is an attempt to mimic natural evolution process in order to exploit new, interesting and not-fully-understood materials for different use cases, most notably for computational purposes.

In this paper, we explore SWCNT-polymer nanocomposite as a stable medium for solving computational tasks. An Evolution-in-Materio platform called Mecobo [2], a product of the EU-funded research project NASCENCE [3], is used to facilitate evolution experiments. Results in this paper show the successful evolution of computationally stable linearly and non-linearly separable logic gates in SWCNTs / polymer materials. Moreover, elementary cellular automata transition functions are also evolved in-materio, providing a new physical abstraction for computation in unconventional materials. The way computation is performed at the physical level is based on local interactions between neighboring units without a central controller. It may be possible to abstract such a process as a ”cellular-automaton-in-materio” and exploit materials to perform computation as a CA, e.g. evolve cellular automata transition tables of different complexity, or even embody a cellular automaton within the material, e.g. exploiting the underlying intrinsic CA-in-materio to execute on a given external input.

The article is laid out as follows: Section II provides background information and Section III describes the experimental setup. In Section IV the results for the evolution of stable logic gates are presented and Section V gives results for the evolution of CA in materio. Discussion and analysis are presented in Section VI and finally Section VII concludes the paper.

II. Background

Evolution-in-Materio is the exploitation of emergent computational behavior in physical materials through artificial evolution. The idea is that some physical processes inherent in different materials may be interpreted as useful computation. EIM attempts to harness this computational power using artificial evolution in order to discover favorable material configurations that may be exploited. One of the earliest attempts at manipulating a physical material for computation was conducted by cybernetician Gordon Pask in 1958. He attempted to create a physical signal processing device based on configurations of grown dendritic iron wires in a ferrous sulphate solution [4]. Modern Evolution-in-Materio was started in 1996 when Adrian Thompson demonstrated that unconstrained evolution in a physically implemented logical system was able to exploit physical properties outside of the logical domain to improve fitness, and hence perform computation [5]. In his experiments, Thompson tried to use artificial evolution to configure a field-programmable gate array (FPGA) to perform tone discrimination. Upon inspection of solutions, it became clear that the computation was not entirely performed in the discrete logical circuit. Instead, the computation relied on physical properties of the FPGA chip itself, outside of the discrete logical domain. This result shows that using evolution to design physically-implemented computers allows for a much larger design space than what is possible with traditional top-down engineering. This, in turn, opens the door to creating more efficient computational devices that to a greater extent exploit natural physical behaviors of the underlying computational substrate.

One challenge in the field of Evolution-in-Materio is discovering which materials are suitable for use as a computational substrate. A good material should preferably exhibit a number of properties that both enable computation and configuration so that evolution can be reliably performed. These properties include having a complex, practically (read: electronically) configurable semi-conducting structure so that the material responds almost instantly and consistently to different inputs, as well as being robust to changes in the external environment such as lighting conditions, temperature and electromagnetic fields [6, 7]. It also helps if the material is easily available. Some interesting candidate substrates that are currently under research are single-walled carbon nanotubes, liquid crystal matrices and silicon FPGA chips.

Single-walled carbon nanotubes are cylindrical carbon allotropes with remarkable physical properties. They exhibit extraordinary electrical properties, which is interesting from an Evolution-in-Materio standpoint. Single-walled carbon nanotubes are engineered by wrapping a one-atom-thick sheet of graphene into a tube. The ”angle” at which the nanotube is wrapped affects the electrical properties of the nanotube - some SWCNTs' show metallic (electrical) conductivity, whilst others show different levels of semiconducting behavior. One way of using SWCNTs for computation is by arbitrarily arranging many SWCNTs in a random network and treating it as a single computational device [8]. The advantage of this is that it enables SWCNT mesh device production at large scale at the wafer level [9]. Single-walled carbon nanotube mesh devices have successfully been used as a substrate for computation in several experiments [10, 11, 12, 13, 14].

The EU project NASCENCE [3, 6], or NAnoSCale Engineering for Novel Computation using Evolution, aims at ”modeling, understanding and exploiting the behavior of evolving nanosystems (e.g. networks of nanoparticles, carbon nanotubes) with the long-term goal to build information processing devices exploiting these architectures without reproducing individual components”. One of the products that have emerged from the NASCENCE project is the Mecobo platform. It is a hardware and software platform for Evolution-in-Materio developed by Lykkebø et al. [2]. It is designed to interface with a large variety of materials, allowing Evolution-in-Materio fitness evaluation directly on the responses of a physical substrate. The Mecobo platform is used for the experiments herein, with a similar approach as in [15, 16, 17].

A. Cellular Automata

Cellular automata are abstract discrete n-dimensional dynamical systems that evolve over time. They consist of a graph of locally-connected nodes that each take on one of k discrete states in time step t. The state of a node n in the graph at time step t is given by the states of n's neighboring nodes at time step $t-1$. Each cellular automaton has a rule table describing how to transition from time step to time step.

In the simplest canonical case, a cellular automaton takes on the form of a one-dimensional array of binary cells where each cell has three neighbors: the cell immediately to the left, the cell immediately to the right, and itself, as illustrated in Fig. 2. These specific cellular automata are called Elementary Cellular Automata [18]. There are 256 possible rule table permutations for the Elementary Cellular Automata. Of these 256 automata, 88 are fundamentally inequivalent [19, p.57].

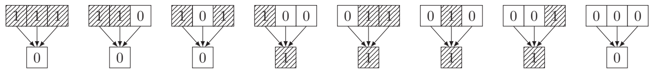


Fig. 1: An Elementary Cellular Automata state transition table laid out to illustrate the Wolfram Code numbering scheme. The top row of groups of three and three cells are the possible neighborhood states for a cell at time step $t-1$, and the cell beneath each group is the resulting state for that same cell at time step t. Reading the bottom row as a binary number ($00011110_2 = 30$) reveals the name of the automaton: Rule 30.

The Elementary Cellular Automata are given a numbering scheme known as the Wolfram Code in [18] that is rooted in binary number representation. Each of the $2^3 = 8$ possible neighborhood states for a given Elementary Cellular Automaton E are represented as each their binary number and ordered numerically. The resulting states for the next time step given from each of these neighborhood states are then taken in order as bits of a new binary number r. This number r is the number identifying E. As an example, Elementary Cellular Automaton Rule 30's rule table and corresponding Wolfram Code number is illustrated in Fig. 1.

B. Cellular Automata as a Physical Abstraction

The big advantage of computation in-materio is that it offers the possibility of performing computation “directly” in the material, as opposed to in some abstracted computational model implemented in a top-down designed computing substrate. The latter will necessarily discard a large part of the computational power of the substrate as a consequence of the abstraction. It seems, then, that exploiting direct in-materio computation is favorable in terms of computation power. However, direct computation in-materio can be quite difficult, especially if universality and scalability is desired. There is an apparent trade-off between efficient usage of the computational complexity in the substrate and ease of programmability for practically useful results which seems to be related to the intuition that computational potential is lost in the abstraction from substrate to theoretical model. The larger the disaffinity between the abstract model and the physical processes in the

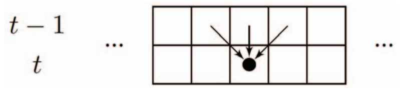


Fig. 2: An example state-time representation of an Elementary Cellular Automaton. The state of a cell is decided by the state of its two neighboring cells and itself in the previous time step.

material becomes, the larger the inefficiency in translation will grow. Thus, finding an abstract computational model that closely matches the physical properties of a material might minimize the computational gap between the physical and the abstract [7].

Cellular automata is a promising abstract computational model for this purpose. Just like they seem to be in physical materials, the computational processes in cellular automata are massively parallel in a distributed and localized fashion. This is a completely different paradigm than the centralized model found in conventional computing.

III. Experiment Setup

In this section, two Evolution-in-Materio experiments are presented. The first experiment attempts to evolve logic gates in the material, as a first stepping stone on the way to cellular automata, to verify that the material is capable of performing simple evolved computation, and to measure the computational stability of the material. The second experiment attempts to evolve Elementary Cellular Automata in the material, to test the central hypothesis that computing with cellular automata in a single-walled carbon nanotube and polymer composite material is an interesting avenue for further research.

The experiments model the material as a binary mapping function capable of mapping a known binary input to a known binary output. The material is assumed to be capable of performing some computational function on a set of inputs coded as voltage patterns applied to one or more electrodes into the material, allowing the output to be read as a voltage pattern from one or more electrodes.

The flow of an experiment is as follows: first, an output calibration over the material is performed for each problem specification. This is done by sweeping through all possible inputs, or at least a somewhat uniformly distributed randomly selected subset of inputs, in order to determine the distribution of possible outputs readable from the material, and to then be able to make a meaningful interpretation of the output. This is necessary because the range of voltages readable on output electrodes to be interpreted as computational output can vary to a great extent based on the way input is coded.

Once an experiment has been calibrated, a Genetic Algorithm solver is run to attempt to find a material configuration that exhibits the desired computational behavior in the material.

Computational stability of a function implemented in-materio is an important concern. Running the same calculation multiple times in-materio should result in a correct output each time – having a computational function only return the correct result a fraction of the time is considerably less useful than one which is consistently correct. Therefore, the computational stability of the evolved configuration is finally tested by repeatedly performing computation in-materio and verifying the correctness of the output.

A. Genetic Algorithm Overview

The genetic algorithm used for the experiments has a population of 40 individuals with generational mixing. Parents are selected by tournament selection where 8 randomly extracted individuals are chosen from the population and, with probability 0.95 the fittest is selected, or else a random individual is chosen. Crossover combines two parents by copying with probability 0.5 each symbol from either the first or second parent. Mutation changes each single gene individually with probability p, where p = 0.01 for the logic gate experiment and p = 0.1 for the CA experiment.

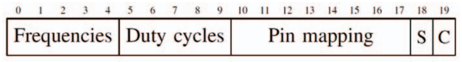


Fig. 3: The genotype mapping for the logic gate experiment. S is the static amplitude of a logical low input signal and C is the center threshold offset

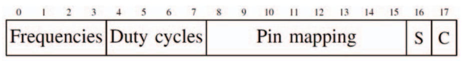


Fig. 4: The genotype mapping for the cellular automata experiment. S is the static amplitude of a logical low input signal and C is the center threshold offset.

The evolvable genotype is represented as a symbol vector of length 20 for the logic gate experiment as represented in Fig. 3, and a vector of length 18 for the CA experiment as shown in Fig. 4. In Fig. 3, the first 5 symbols represent material configuration signals in a form of pulse wave frequencies and the next 5 symbols represent the duty cycles. The next 8 symbols represent the pin mapping. Then follows a symbol representing the static amplitude of logical low coding signal. The last symbol represents an output interpretation threshold offset. The genoype for the CA experiment in Fig. 4 is similar to the one used in the logic gate experiment, except that it has been adjusted for the different number of input and configuration electrodes.

The fitness function for the logic gate experiment is given as a function of how well it can compute the binary computation on the 4 inputs (0,0), (0,1), (1,0), and (1,1). For each set of inputs, the actual output was compared to the expected output. If they matched, 2 points were awarded. If the output was undecided, i.e. the measured output average was within the noise-reducing padding area around the threshold, 1 point was awarded. 0 points were awarded if the output did not match. For the CA experiment, the fitness is a function of how well it can compute the target cellular automaton's state transition function on all the 8 possible input combinations. The squared scaled confidence value

$$c_n = \frac{(o_n - T)^2}{5^2} \tag{1}$$

for input combination n is then calculated for each input combination as defined in (1), where T is the output threshold value, and o_n is the averaged output value for input combination n. For each set of inputs, the interpreted output is compared to the expected output from the input-to-output mapping function. If the output matches, c_n points are awarded to the total fitness. If the output is undecided, 0 points are

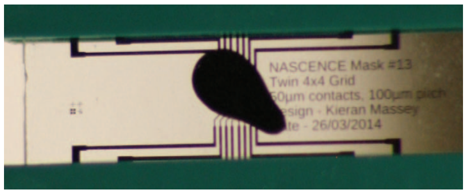


Fig. 5: A photograph of the SWCNT material on its glass slide.

awarded to the total fitness. If the output otherwise does not match, $-10c_n$ points are awarded to the total fitness. Then, if all outputs were correct, an additional 8 points are awarded to the total fitness. Finally, the fitness is transposed by an additional 8 points, and then scaled by a factor of 0.0625 in order to obtain values in the range between 0 and 1+, where a score of 1 (or better) signifies a fully correct evolved solution. The fitness values did not directly account for stability, e.g. by performing the same fitness evaluation multiple times and averaging the result.

B. Material Overview

The computational substrate material used in the experiments is a random mesh of single-walled carbon nanotubes mixed with poly(butyl methacrylate) (PBMA) and dissolved in anisole (methoxybenzene). This is laid out in a glass slide with 16 gold electrodes arranged in a 4x4 grid with contacts of $50\,\mu\mathrm{m}$ in diameter and $100\,\mu\mathrm{m}$ pitch between contacts. The material, slide #1 of batch #15, numbered B15S01, was prepared by Kieran Massey at Durham University by the following method: "20 µl of material are dispensed onto the electrode area; This is dried at 85°C for 30 min to leave a *thick film*; The hotplate is turned off and the substrates are allowed to cool slowly over a period of roughly 2 h to room temperature." After this process, the CNTs are not movable. The carbon nanotube concentration in the material is 0.75% by weight. All electrodes show connection resistances on the order of $20\,\mathrm{k}\Omega$, but it is reasonable to assume that the nanotube coverage over the electrodes is noticeably uneven, given the nanotube concentration level [20]. Fig. 5 shows a photograph of the material on its glass slide. The material is produced within the NASCENCE project [6].

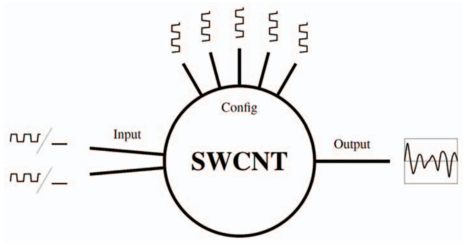


Fig. 6: Example of input, output and configuration mapping of the material interface electrodes for logic gate computation.

TABLE I: Logic Gates Input-to-Output-Mapping Function

I_1	I_0	O_{AND}	O_{OR}	O_{XOR}
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

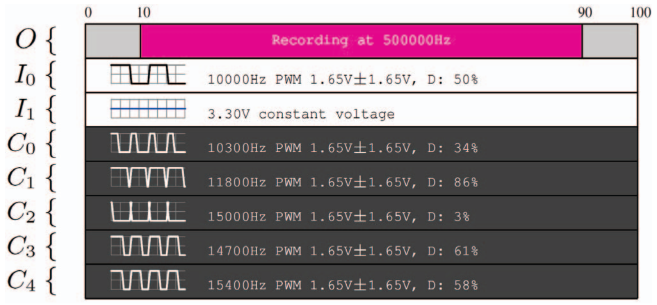
IV. Evolving Stable Logic Gates

This first experiment attempts to evolve Boolean binary logic gates in the material to see if the material is capable of performing simple evolved computation, and to measure the computational stability of the material. Three Boolean logic gates were evolved: the AND gate, the OR gate, and the XOR gate. The first two gates are implementations of linearly separable Boolean functions, whilst the third is not. The input-to-output-mapping function for each of the three evolved gates can be seen in TABLE I.

Seven electrodes into the SWCNT/polymer composite material were used as input electrodes. Two of these electrodes represent the two inputs a and b to the binary Boolean function. Each of these two inputs a and b can be either a logical 0 or a logical 1. A logical 0 on an input electrode is realized by applying a selected constant static voltage, the exact value of which is decided on a per-solution basis. A logical 1 on an input electrode is represented by by applying a digital square wave with a frequency of 10000Hz and a $3.3\,\mathrm{V}$ amplitude, from $0\,\mathrm{V}$ to $3.3\,\mathrm{V}$. The other five input electrodes are used as material configuration inputs, with a digital pulse wave with a specific frequency and duty cycle chosen on a per-solution basis from a predetermined range being applied upon each of them.

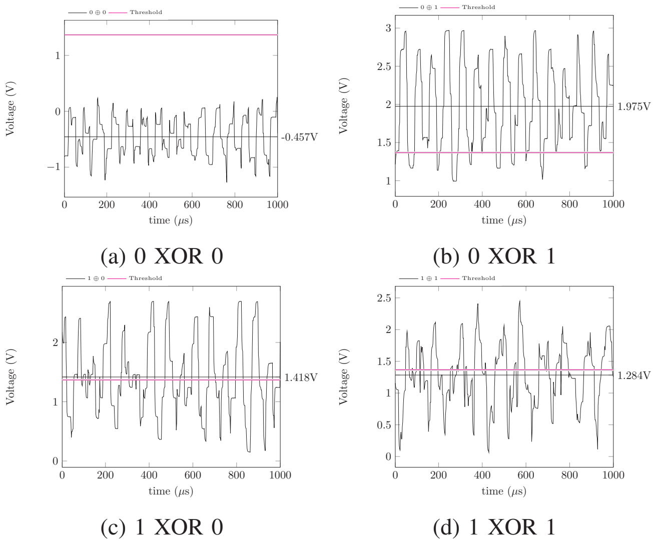
The computational output expected from a Boolean logic gate is a single Boolean value: 0 or 1. A single electrode output electrode was used. The voltage output was sampled at 500000Hz for 80 ms from the 10th to the 90th ms of computation. The samples were averaged arithmetically and compared to a predetermined threshold value. If the average was higher than the threshold value plus some small empirically determined padding to reduce measurement noise, the output was interpreted as a logical 1. Conversely, if the average was lower or equal to the padded threshold value, the Boolean output was interpreted as a logical 0. The threshold value was obtained experimentally by running a calibration sweep of the material, which consisted of performing 200 logic gate computations with randomly generated inputs conforming to the defined input coding. The averaged output was calculated for each computation, and the median of these averages were taken to be the threshold value.

Which specific physical electrodes were used for which of the input and output signals was decided by a logical-to-physical mapping of electrodes coded in the evolvable genotype. A schematic view of the SWCNT/polymer composite material with its logical input and output electrodes can be seen in Fig. 6.



0
10
90
100
O
I_0
I_1
C_0
C_1
C_2
C_3
C_4
Recording at 500000Hz
10000Hz PWM 1.65V±1.65V, D: 50%
3.30V constant voltage
10300Hz PWM 1.65V±1.65V, D: 34%
11800Hz PWM 1.65V±1.65V, D: 86%
15000Hz PWM 1.65V±1.65V, D: 3%
14700Hz PWM 1.65V±1.65V, D: 61%
15400Hz PWM 1.65V±1.65V, D: 58%

Fig. 7: A timing overview of the activity on each logical electrode over time during a single computation of $1 \oplus 0$ on the evolved xor gate. Time progresses along the x-axis, labeled in milliseconds at points of interest. O is the output pin, I_n are the input pins, and C_n are the configuration pins. The small waveform illustrations reflect the duty cycle, but not the frequency. D is the duty cycle.



(a) 0 XOR 0 (b) 0 XOR 1 (c) 1 XOR 0 (d) 1 XOR 1

Fig. 8: The first 500 raw material output samples recorded during four XOR computations. The horizontal black lines represent average values for a sample series, and the horizontal red lines represent the output interpretation threshold value, which is 1.367V.

A. Results

Computationally stable AND, OR and XOR logical gates were successfully evolved in SWCNT materials. The evolved results herein show that it is possible to obtain a full range of logical functions that are also stable and show predictable behavior. Many different configurations yielding different logic gates were found. One configuration for each of the three types of logic gate is here selected for further examination.

Although the search space in this experiment is quite large, there seems to be a lot of satisfactory solutions amongst the candidates. The search space contains 256^{20} possible candidates, yet in a typical Genetic Algorithm run with a population size of only 40 individuals, the solver is able to frequently find satisfactory solutions in the first few generations. Three evolved solutions were tested for correctness by performing repeated computations in-materio over all four possible inputs. Each solution was tested for 2000 repeated calculations of each of the four possible inputs, for a total of 8000 tests. The evolved OR gate computed the correct value 8000 out of 8000 times. The evolved AND gate also computed the correct value 8000 out of 8000 times. The evolved XOR gate computed the correct value 7955 out of 8000 times, resulting in an estimated error rate of $0.5625\% \pm 0.1639$ *pp* at a 95% level of confidence.

Here, a set of recorded XOR computations is presented. Fig. 7 shows the state of the (logically mapped) input and output electrodes over the course of the execution of the computation of the logical task $1 \oplus 0$. Fig. 8 shows samples recorded from four XOR calculations performed in-materio. The entire recording for each calculation is in reality 40000 samples long, but for practical reasons only the first 500 samples are plotted here. The horizontal black lines represent the measured total average of the entire recording for each calculation. The horizontal magenta line together with a small gray padding band represents the threshold value above or under which a recorded average is taken to be a logical 1 or 0, respectively.

TABLE II: Input-to-Output-Mapping Functions For the Elementary Cellular Automata

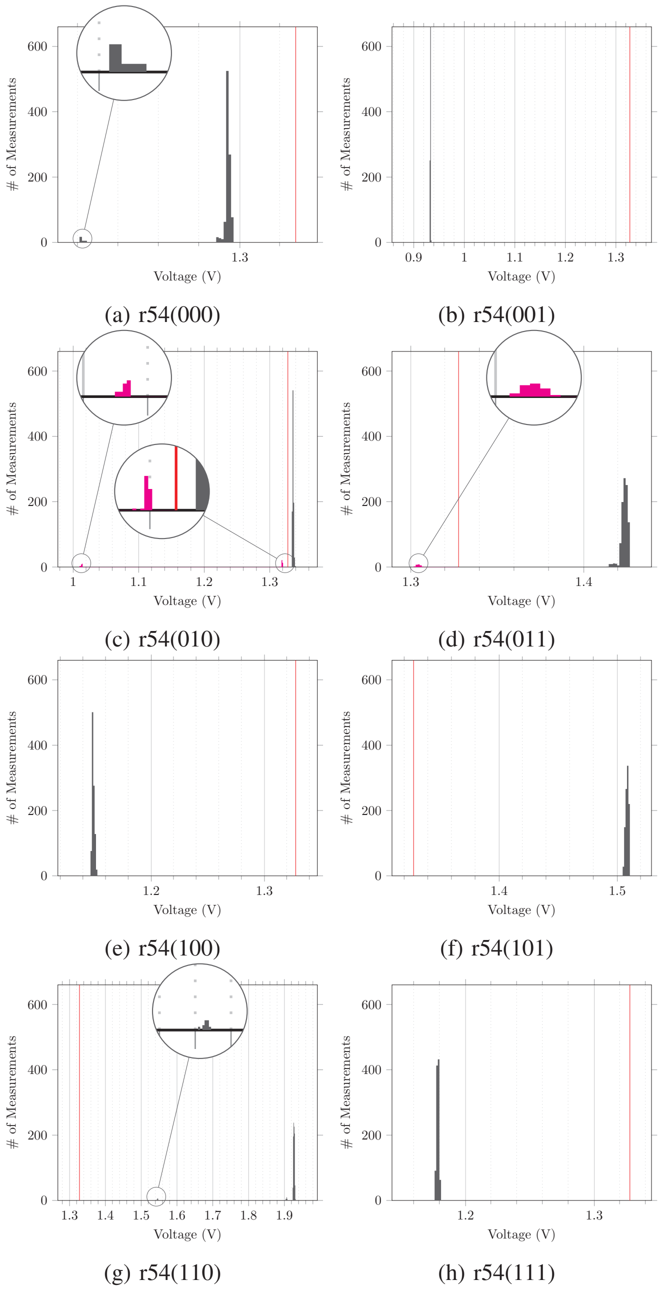
I_2	I_1	I_0	$O_{\text{Rule 54}}$	$O_{\text{Rule 151}}$
0	0	0	0	1
0	0	1	0	1
0	1	0	1	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	0	1

V. Evolving Elementary Cellular Automata

This experiment attempts to evolve Elementary Cellular Automata transition functions in-materio, to test the central hypothesis that computing with cellular automata in a single-walled carbon nanotube and polymer composite material is an interesting avenue for further research. Of particular interest are the Elementary Cellular Automata in Class III and Class IV of Wolfram's Elementary Cellular Automata classification, the former because they can be capable of performing complex computations [21], and the latter because they are famously conjectured by Wolfram to be capable of universal computation [19]. Two stable Elementary Cellular Automata have been evolved: Rule 151, a chaotic Class III cellular automata, and Rule 54, a complex Class IV cellular automata conjectured but not yet proven to be computationally universal [19, 22].

Like in the logical gate experiment, the computational function can be thought of as a mapping from binary inputs to a binary output. The expected input-to-output mappings for

Rule 54 and Rule 151 are given in TABLE II, where I_n is input n, and $O_{\text{Rule } n}$ is the output for Rule n. The values are given in the logical domain.



(a) r54(000)
(b) r54(001)
(c) r54(010)
(d) r54(011)
(e) r54(100)
(f) r54(101)
(g) r54(110)
(h) r54(111)

Fig. 9: Histograms representing the measured average outputs over 8000 calculations using the evolved Rule 54 configuration, grouped by input. The output threshold used for mapping the measured output into the binary logical domain is plotted as a red vertical line in each histogram. Measurements that result in a wrong computational answer are in magenta. Interesting details are magnified.

Just as in the logic gate experiment, seven input electrodes were used. This time, three of these electrodes represent the state of the three cells in the neighborhood set for an elementary cellular automaton rule transition. Each of these three inputs can either be a logical 0 or a logical 1, coded as a static voltage or a digital pulse applied to the input electrodes, respectively. The parameters for the static voltages and digital pulses are the same as in the logic gate experiment. The other four input electrodes function as a static material configuration, each electrode being applied upon a digital pulse signal chosen from a specific range of frequency and duty cycle combinations, like in the logic gate experiment.

The computational output expected from any binary cellular automaton transition function is a single Boolean value: 0 or 1. This is the same output range as can be expected from a binary logical gate. Therefore, the output interpretation scheme for this experiment is the same as the one in the logic gate experiment, recalibrated for the input coding for the CA over 200 computations.

A. Results

Computationally stable Rule 54 and Rule 151 transition functions have been evolved. The Rule 151 solution was found in the 10th generation of the Genetic Algorithm. The Rule 54 solution was found in the 5th generation of the Genetic Algorithm.

Both solutions were tested for correctness by performing repeated computations in-materio over all eight possible inputs. Each solution was tested for 1000 repeated calculations of each of the eight possible inputs, for a total of 8000 tests. The evolved Rule 54 computed the correct value 7917 out of 8000 times, resulting in an estimated failure rate of $1.0375\% \pm 0.2221$ *pp* at a 95% level of confidence. The evolved Rule 151 computed the correct value 8000 out of 8000 times. Fig. 9 shows the levels of each of the measured average outputs for each of the 8000 calculations grouped by input.

Here, a set of recorded Rule 54 computations is presented. Fig. 10 shows the state of the (logically mapped) input and output electrodes over the course of the execution of the computation of Rule 54 transition function for the neighborhood state 011_2. Fig. 11 shows samples recorded from eight Rule 54 calculations performed in-materio. Again, like with the logic gates, the entire recording for each calculation is in reality 40000 samples long, and again for practicality reasons only the first 500 samples are plotted here. The horizontal black lines represent the measured total average of the entire recording for each calculation. The magenta threshold together with a small gray padding band represents the threshold value above or under which a recorded average is taken to be a logical 1 or 0, respectively.

VI. Discussion

The single-walled carbon nanotube mesh and polymer composite material used in these experiments has shown itself capable of performing complex non-linearly separable computation tasks. In the following subsections, several computational aspects are discussed and analyzed.

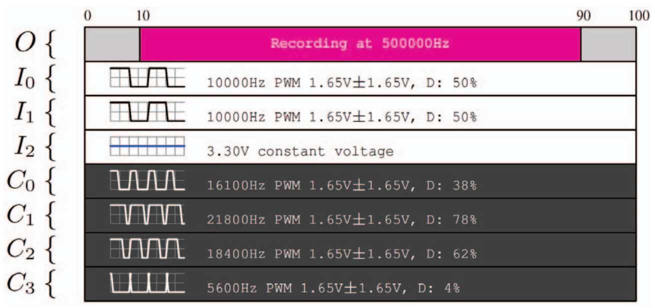

0
10
90
100
O
I_0
I_1
I_2
C_0
C_1
C_2
C_3
Recording at 500000Hz
10000Hz PWM 1.65V±1.65V, D: 50%
10000Hz PWM 1.65V±1.65V, D: 50%
3.30V constant voltage
16100Hz PWM 1.65V±1.65V, D: 38%
21800Hz PWM 1.65V±1.65V, D: 78%
18400Hz PWM 1.65V±1.65V, D: 62%
5600Hz PWM 1.65V±1.65V, D: 4%

Fig. 10: A timing overview of the activity on each logical electrode over time during a single computation of r54(011_2) on the evolved Rule 54 device. Time progresses along the x-axis, labeled in milliseconds at points of interest. O is the output pin, I_n are the input pins, and C_n are the configuration pins. The small waveform illustrations reflect the duty cycle, but not the frequency. D is the duty cycle.

A. *Stability of Results*

The stability of the results is greater than that of previous work [10, 2], and the evolved solutions seem stable enough to be called "stable" solutions in the context of Evolution-in-Materio. Looking closer at the distribution of measurements in Fig. 9, a peculiarity becomes apparent. There is sometimes a small separate clustered group of measurements far away from the median which severely reduce the stability of the otherwise very tight clustering of measurements around the median. The same effect was also observed for the XOR computation experiment. This may be caused by some complex intrinsic process within the material itself, but it may also be caused by some experimental error in process, equipment, software or similar. If the latter is the case, the true computational stability of the solutions may very well be much greater than what they are measured to be in the experiments in this paper. Note that tests with longer time intervals between subsequent measurements, i.e., relaxation time, did not infulence the results. Still, how stable is stable enough? Comparing to the error rate of consumer-grade conventional computers, which, while not published anywhere, seems to be on the order of one quintillion operations per error[1], the computational devices presented in this paper are anything but stable. TABLE III shows an overview of the stability and error rates for the experiments presented in this paper.

B. *Speed of Computation*

Currently, performing a computation in-materio takes on the order of $100\,\mathrm{ms}$ to complete. This is because the input/output encoding is specified somewhat arbitrarily to last for that length of time. $100\,\mathrm{ms}$ is quite slow compared to even consumer-grade conventional computers, which are easily capable of up to billions of operations over the same time period. That being said, the results presented in this paper are a proof of concept, and computation speeds may be improved upon in further work.

[1]An estimated 2 billion operations each second every day for 20 years before the silicon microchip wears out.

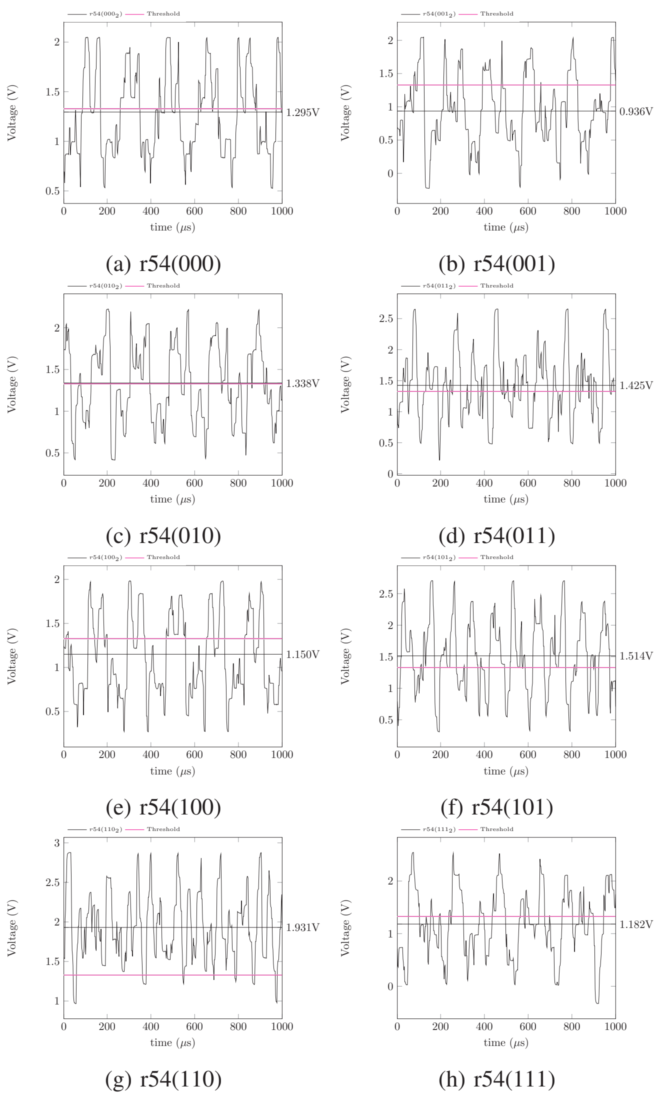


Fig. 11: The first 500 raw material output samples recorded during eight Rule 54 computations. The black horizontal lines represent average values for a sample series, and the red horizontal line is the output threshold, which is 1.328V.

C. *Computational Capacity of the Material*

Does the computation actually take place in-materio? When performing evolution-guided search for computation in a material, the entire input domain and output range of the computational function is known, and a signal encoding and decoding process is performed off-material. This can make it hard to pinpoint exactly where the computation takes place. Certainly it is possible to construct a fitness evaluator and input/output encoding that is so complex that it can find computation in anything – even random noise. In such a case, the computation is in reality happening outside of the material. How can the origin of computation be measured? It can be helpful to

TABLE III: Summary of the stability of result and error rates (at 95% level of confidence) for the experiments presented in the paper

Function	Correct tests	Total tests	Error rate
AND	8000	8000	0
OR	8000	8000	0
XOR	7955	8000	$0.5625\% \pm 0.1639$ *pp*
Rule 54	7917	8000	$1.0375\% \pm 0.2221$ *pp*
Rule 151	8000	8000	0

replace the material with different hypothetical materials and imagine what would happen if the same computations were performed using the switched hypothetical materials, but still using the same input and output coding schemes. Considering the following three hypothetical alternative materials, some insight might be gained into the computational complexity of the SWCNT material:

1) a computationally "dead" material that always outputs the same static signal(s);
2) a material that produces "true random noise" on its output(s) regardless of the input; and
3) a material that linearly combines its input(s) and passes it on to its output(s).

Does the computation that allegedly happens in the real material also happen when the material is replaced with one of these hypothetical materials? For one, the real material certainly outperforms the "dead" material – all of the implemented functions show a range that requires the output voltage to be above or below some static non-changing threshold level depending on the input. Since the expected output depends on the input, and the static threshold crucially does *not* change based on the input, it demonstrably performs more computation than the "dead" material.

Now, in the case of the random material, it is possible that the random output happens to measure on the right side of the threshold level for different inputs by pure chance. However, it will probably not do so very often, statistically speaking. The evolved devices presented in this paper are all reasonably stable in their output, or at least much more stable than what one can expect from a "true" random material. This indicates that at least some of the computation happens in the material itself.

In the case of the linearly combining material, linear computation is possible in-materio almost by definition, but computations that are not linearly separable should not be implementable. The evolved XOR gate, and the two evolved Elementary Cellular Automata in the SWCNT material, however, are not linearly separable functions. Thus, the SWCNT material exhibit computational promise beyond linearly separable functions.

VII. Conclusion

Computationally stable linearly and non-linearly separable logic gates, and stable non-linearly separable Class III and Class IV Elementary Cellular Automata state transition calculators have been evolved in a single-walled carbon nanotube and polymer mesh substrate. This was done as a first step in exploring the possibility of implementing cellular automata as a computational abstraction over carbon nanotube substrates. Ultimately, the goal is to map the feasibility of meaningful computation in an SWCNT material using cellular automata as the principal computational abstraction tool. In addition, stability of solutions has been analyzed, together with a discussion on the speed of computation and computational capacity of materials.

Building upon the methods developed and results achieved in this paper, one possible avenue for further work is to continue evolving more complex and more computationally powerful Elementary Cellular Automata in-materio. An obvious target is Rule 110, since it has been shown to be capable of universal computation by way of simulating a specific universal cyclic tag system [23]. Once the rule table for Rule 110 is successfully evolved in-materio, a practical simulation of the Rule 110 cellular automaton will be possible using a hybrid conventional/materio approach where the state of the simulation is kept track of on a traditional computer for practicality, and transitions from state to state in the cellular automaton are calculated by querying the material.

Another interesting avenue for further investigation is the relationship between the computational capacity of single-walled carbon nanotube meshes, cellular automata classifications and parametrizations, and suitability for evolution in-materio. One possible experiment is to attempt to evolve a multitude of different cellular automata in-materio, and compare the difficulty of evolution, stability of the solution, and the classification of the cellular automata evolved. Are cellular automata which exhibit interesting computational properties more difficult to evolve? Easier to evolve? What is the relationship between evolvability and the λ-parameter [24]? There are a lot of questions yet to be answered.

VIII. Acknowledgements

The research leading to these results has received funding from the European Commission's Seventh Framework Programme (FP/2007-2013) under grant agreement 317662.

References

[1] J. F. Miller and K. Downing, "Evolution in materio: Looking beyond the silicon box," in *Evolvable Hardware, 2002. Proceedings. NASA/DoD Conference on*, pp. 167–176, IEEE, 2002.

[2] O. Lykkebø, S. Harding, G. Tufte, and J. Miller, "Mecobo: A Hardware and Software Platform for In Materio Evolution," in *Unconventional Computation and Natural Computation* (O. H. Ibarra, L. Kari, and S. Kopecki, eds.), vol. 8553 of *Lecture Notes in Computer Science*, pp. 267–279, Springer International Publishing, 2014.

[3] H. Broersma, F. Gomez, J. F. Miller, M. Petty, and G. Tufte, "Nascence project: Nanoscale engineering for

novel computation using evolution," *International Journal of Unconventional Computing*, vol. 8, no. 4, pp. 313–317, 2012.

[4] G. Pask, "Physical analogues to the growth of a concept," *Mechanisation of Thought Processes: Proceedings of a Symposium Held at the National Physical Laboratory.*, pp. 879–922, 1958.

[5] A. Thompson, "An Evolved Circuit, Intrinsic in Silicon, Entwined with Physics," in *Proc. 1st Int. Conf. on Evolvable Systems (ICES'96)* (T. Higuchi, M. Iwata, and L. Weixin, eds.), (Berlin), pp. 390–405, Springer-Verlag, 1997.

[6] "NASCENCE Project Website, http://nascence.no." [Accessed 28-May-2015].

[7] S. Stepney, "The Neglected Pillar of Material Computation," *Physica d-Nonlinear phenomena*, vol. 237, no. 9, pp. 1157–1164, 2008.

[8] E. S. Snow, J. P. Novak, P. M. Campbell, and D. Park, "Random networks of carbon nanotubes as an electronic material," *Applied Physics Letters*, vol. 82, no. 13, pp. 2145–2147, 2003.

[9] J. Gabriel, K. Bradley, and P. Collins, "Dispersed growth of nanotubes on a substrate," May 10 2006. EP Patent App. EP20,030,808,389.

[10] A. Kotsialos, M. Massey, F. Qaiser, D. Zeze, C. Pearson, and M. Petty, "Logic gate and circuit training on randomly dispersed carbon nanotubes.," *International journal of unconventional computing.*, vol. 10, pp. 473–497, September 2014.

[11] M. Mohid, J. Miller, S. Harding, G. Tufte, O. Lykkebo, M. Massey, and M. Petty, "Evolution-in-materio: A frequency classifier using materials," in *Evolvable Systems (ICES), 2014 IEEE International Conference on*, pp. 46–53, Dec 2014.

[12] M. Mohid, J. Miller, S. Harding, G. Tufte, O. Lykkeb, M. Massey, and M. Petty, "Evolution-In-Materio: Solving Machine Learning Classification Problems Using Materials," in *Parallel Problem Solving from Nature PPSN XIII* (T. Bartz-Beielstein, J. Branke, B. Filipi, and J. Smith, eds.), vol. 8672 of *Lecture Notes in Computer Science*, pp. 721–730, Springer International Publishing, 2014.

[13] M. Mohid, J. Miller, S. Harding, G. Tufte, O. Lykkebo, M. Massey, and M. Petty, "Evolution-in-materio: Solving function optimization problems using materials," in *Computational Intelligence (UKCI), 2014 14th UK Workshop on*, pp. 1–8, Sept 2014.

[14] K. Clegg, J. Miller, K. Massey, and M. Petty, "Travelling Salesman Problem Solved in materio by Evolved Carbon Nanotube Device," in *Parallel Problem Solving from Nature PPSN XIII* (T. Bartz-Beielstein, J. Branke, B. Filipi, and J. Smith, eds.), vol. 8672 of *Lecture Notes in Computer Science*, pp. 692–701, Springer International Publishing, 2014.

[15] O. R. Lykkebø, S. Nichele, and G. Tufte, "An investigation of square waves for evolution in carbon nanotubes material," in *13th European Conference on Artificial Life*, Springer, 2015.

[16] S. Nichele, O. R. Lykkebø, and G. Tufte, "An investigation of underlying physical properties exploited by evolution in nanotubes materials," in *Proceedings of 2015 IEEE Symposium Series on Computational Intelligence, SSCI 2015*, IEEE, 2015.

[17] S. Nichele, D. Laketic, O. R. Lykkebø, and G. Tufte, "Is there chaos in blobs of carbon nanotubes used to perform computation?," in *7th International Conference on Future Comp. Tech. and Applications*, XPS Press, 2015.

[18] S. Wolfram, "Statistical mechanics of cellular automata," *Rev. Mod. Phys.*, vol. 55, pp. 601–644, Jul 1983.

[19] S. Wolfram, *A New Kind of Science*. Wolfram Media, 2002.

[20] K. Massey, "Material for NTNU 3." private communication, 2014.

[21] G. J. Martinez, J. C. Seck-Tuoh-Mora, and H. Zenil, "Wolfram's Classification and Computation in Cellular Automata Classes III and IV," *ArXiv e-prints*, 2012.

[22] G. J. Martínez, A. Adamatzky, and H. V. McIntosh, "Phenomenology of glider collisions in cellular automaton rule 54 and associated logical gates," *Chaos, Solitons & Fractals*, vol. 28, no. 1, pp. 100–111, 2006.

[23] M. Cook, "Universality in Elementary Cellular Automata," *Complex Systems*, vol. 15, no. 1, pp. 1–40, 2004.

[24] C. Langton, "Computation at the Edge of Chaos Phase Transitions and Emergent Computation," *Physica D*, vol. 42, p. 1237, June 1990.