

Is Spiking Logic the Route to Memristor-Based Computers?

(Invited Paper)

Ella Gale

Unconventional Computing Centre
Bristol Robotics Laboratory
University of the West of England
Bristol, UK BS16 1QY
Email: ella.gale@uwe.ac.uk

Ben de Lacy Costello

Unconventional Computing Centre
Faculty of Applied Sciences
University of the West of England
Bristol, UK BS16 1QY
Email: ben.delacycostello@uwe.ac.uk

Andrew Adamatzky

Unconventional Computing Centre
Computer Science and Creative Technology
University of the West of England
Bristol, UK BS16 1QY
Email: andrew.adamatkzy@uwe.ac.uk

Abstract—Memristors have been suggested as a novel route to neuromorphic computing based on the similarity between neurons (synapses and ion pumps) and memristors. The D.C. action of the memristor is a current spike, which we think will be fruitful for building memristor computers. In this paper, we introduce 4 different logical assignments to implement sequential logic in the memristor and introduce the physical rules, summation, ‘bounce-back’, directionality and ‘diminishing returns’, elucidated from our investigations. We then demonstrate how memristor sequential logic works by instantiating a NOT gate, an AND gate and a Full Adder with a single memristor. The Full Adder makes use of the memristor’s memory to add three binary values together and outputs the value, the carry digit and even the order they were input in.

I. INTRODUCTION

Memristors have been compared to both neurons [1], [2] and synapses [3]–[5] and have widely been anticipated as a useful route towards neuromorphic (brain-like) computing due to the memristor’s ability to hold a memory or state [6]. Although the memristor was predicted to exist based on symmetry concerns [6] real world analogues were not recognised to exist until 2008 [7], even though they were made before [8]. The memristor is commonly considered as an A.C. element [9], however the D.C. response of the memristor is highly interesting because memristors possess spike-like dynamics [10] which have been shown to combine in spike-train-like ways in memristor networks [11]. Furthermore, the interactions of these spikes, which can be considered the short-term memory of the memristor, can be used with a novel sequential logic approach which can be used to make simple logic circuits [12].

The advantages of using spike interactions are many-fold. The memristor switching itself can be slow [13] but the spikes can interact much faster, the output of which is ‘held’ in the short-term memory of the memristor, which gives rise to, if not faster processing, more complex operations within a given time-frame than is usually the case in standard electronics.

When designing devices to compute with a certain logic, we have some freedom in how the logical ‘1’s and ‘0’s are assigned. For these devices we shall take the voltage inputs as the logical input, with the current values as the logical output, where it is understood that some processing (via a

memristor or other circuitry) is required in order to instantiate the logical circuit. Within this assumption, it is possible to compute operations of a surprisingly high complexity with just a single memristor, and, it is suggested, that with this approach and the conversion circuitry, a useful approach to computation.

In this paper we will examine in more detail the physics of the memristors and the physical rules in order to understand the operation of sequential logic and present some examples of the high-level computation that a single memristor can obtain.

II. METHODOLOGY

A. Sequential Logic

We shall make use of sequential logic, as implemented in [12], which works with the spike interactions seen in the memristor. Memristor sequential logic allows the computation through time by storing a state and allowing it to interact with the input; thus a one terminal device can do two-input (or higher) logical operations, if we are willing to wait for the output.

As shown in figure 1 the memristor’s state is stored in its short-term memory and the current output to a voltage change is actually a function of its zeroed/null state and the input. Sequential logic makes use of the memristor’s short-term memory to store the first bit, A , of an operation before the transmission of the second bit B . The output at time, t_A , is a function of A and the memristor’s starting state (which is \emptyset if the device has been properly zeroed), given by $f(A, \emptyset)$. At time t_B (where t_B is one measurement step after t_A) the output would be $f(B, A)$. The response step, t_1 is measured one measurement step after t_b . Thus, this voltage data is input at t_A and t_B and measured at t_1, t_2 and so on where $t_a < t_b < t_1 < t_2$.

B. Experimental

Memristors were fabricated as in [13] using the TiO_2 sol-gel as described in [14] and were measured using a Keithley electrometer, with a set time-step of 0.1s, which gives an actual output of 0.16s. After each logical test, the memristor was left for 40 timesteps ($\sim 40\text{s}$) to lose its short-term memory, i.e.

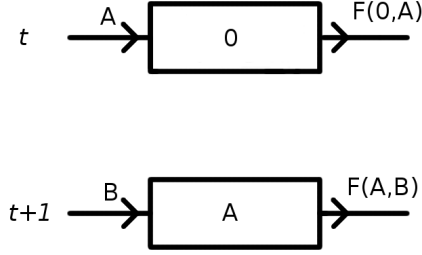


Fig. 1. Sequential Logic. The output of the memristor is a function of its state, as shown in the box, and the input. As the state is stored for the duration of the short-term memory logical values can be combined if they are input sequentially.

TABLE I
FOUR DIFFERENT METHODS OF IMPLEMENTING LOGICAL ‘1’ AND ‘0’
WITH MEMRISTOR SPIKES: M REFERS TO A HIGH MAGNITUDE VOLTAGE,
 m TO A LOW MAGNITUDE VOLTAGE AND ‘+’ AND ‘-’ REFER TO ITS
POLARITY.

Logical value	Magnitude Logic	Polarity Logic	Mixed Logic 1	Mixed Logic 2
One	M	+	$+M$	$-M$
Zero	m	-	$-m$	$+m$

reset to the null state. All presented results are experimental data.

There are two variables we can utilise when assigning logical values: the magnitude, as represented by M for a high magnitude and m for a low magnitude; and the sign, as represented by a $+$ for positive and $-$ for negative. The 4 different logical assignments that can be applied using these values is shown in table I. To implement logical operations, voltage spikes are applied for one time-step and the response recorded at the same frequency. In between logical operations, the devices were left for longer than the equilibration time (τ_∞ in [10] which is around 3.5s) to zero the memristor by removing its short term memory.

III. ELUCIDATED RULES

Changing the values of M and m can allow the results to be tuned or balanced against the effect of polarity, but in this paper we shall just deal with qualitative examples. From investigation of these systems, we have elucidated the following physical rules for the system.

A. Directionality

The memristor naturally implements Implication (as first invented by Bertrand Russell and observed in [15]). The memristor is directional: e.g. The response at t_1 , for $A \rightarrow B$ does not equal the output (t_1) for $B \rightarrow A$. The cause for this is that the memristor responds to the difference in voltage. This naturally allows memristor-based sequential logic to compute implication logic as Implication, IMP or \rightarrow , requires that $0 \rightarrow 1 \neq 1 \rightarrow 0$ and thus the order in which the two values are input has a meaning. Naturally, sequential logic,

as it separates the values in time, implements this ordering. Note that sequential logic is a scheme for how the memristor can enact logical operations, Implication is an example of a logical operation, and implication logic is the name for the logical set of [IMP, FALSE] required for functionally-complete computation.

B. ‘Summation’ via Energy Conservation

If the logical ‘1’ is taken as being a high voltage, i.e. M instead of m , then more energy is imparted to the system from the logical combinations like [1,1] compared to [0,0]. This approach can allow the creation of memristor based time-limited summators of use in leaky integrate and fire neurons.

C. ‘Bounceback’

The application of a voltage spike produces a resultant current spike in the direction of the difference between the starting voltage and the ending voltage, e.g. the first voltage change $V_0 \rightarrow V_A$ causes a positive current response, $+i_A$, if V_A is positive, and negative, $-i_A$, if V_A is negative. If the system is then returned to zero, there is a smaller current spike of the opposite polarity, i.e. $-i_0$ and $+i_0$ respectively for the two examples mentioned above. If several spikes are input before returning to zero, i.e. a sequence of $[V_0, V_A, V_A, V_0]$ the current spike is larger than would be the case for $[V_0, V_A, V_0]$, although not twice as large due to losses in the system.

D. ‘Diminishing Returns’

The effect of additional spikes of the same size and polarity, occurring within the window of the memristor’s short term memory decreases. A similar effect is seen with changing polarity, in that changing polarity can cause a larger response (than not) but this response is smaller with successive voltages, i.e. the response spike to $[V_0, +V_A, -V_A, +V_A, -V_A]$ is smaller than $[V_0, +V_A, -V_A]$.

IV. EXAMPLES OF LOGICAL SYSTEMS

Knowledge of these rules and effects allows us to design logical computation systems which perform a surprising amount of computation with only a single memristor. We have found that, in these schemes that the summation effect is important in magnitude logic, the ‘bounceback’ effect is more relevant in polarity logic (although both affect the outcome). As these can be balanced and set in opposition to each other, the richest effects came from using the mixed logics (as presented in I: we will now present a few examples.

A. Inverter

Using polarity logic and the ‘bounce-back’ effect, it is an easy thing to build an inverter as shown in figure 2. Because the response spike is in the opposite direction, taking that as the result of the operation switches from 1 to 0 (and vice versa) and can be viewed as performing the NOT operation on the input.

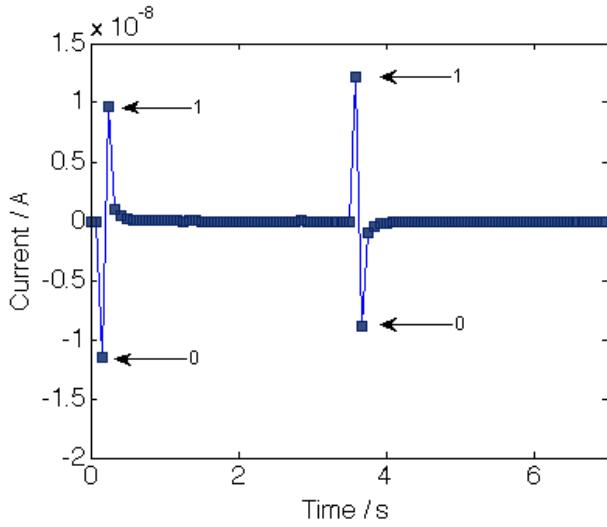


Fig. 2. NOT Gate Implementation

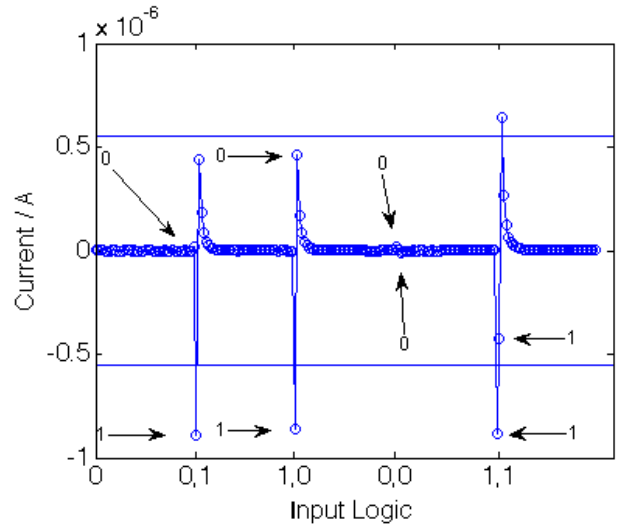


Fig. 3. AND Gate Implementation

B. AND Gate

An example of an AND gate is shown in figure 3, this example uses mixed logic 2 with a M of $-0.5V$ and a m of $+0.001V$. If we take the response output as '1' if current over a threshold (in this case, $0.55\mu A$) is seen, the device implements an AND gate (this is still the case if we choose to limit ourselves to only the value of the t_1 response spike). Due to the summation effect, the amount of energy in the [1,1] system is larger than the [0,1], [1,0] and [0,0] parts of the truth-table, and this causes a larger 'bounceback' response which can be measured in the positive current response.

Were we to limit ourselves to the negative current part of the device response, the magnitude of the output picks out an inclusive OR operation, in that the only parts of the truth table that have a response over the threshold are those that contain a '1' (because these spikes depend on a '1' input). Although this response is trivial, it is information that can be usefully used with the correct output circuitry.

C. Towards a Full-Adder

It is possible to compute an unconventional instantiation of full-adder, as shown in figure 4 (admitting that we require a voltage spike to current spike conversion). The two input and carry bits are input as a series of spikes using mixed logic 2 with input '1' represented by $-0.5V$ and input '0' represented by $+0.001V$. The input sequence is [A,B,C,1,2,3,4], with the logic input at $t_A - t_C$, the response spike recorded at t_1 , an extra read voltage of $-0.15V$ input at t_2 . This gate requires a clock to operate. Figure 4 shows the response of the memristor to this scheme, for the three inputs of a full adder, the read spike at t_2 is marked with an * to make it easier to understand, and the data of the memristor losing its short-term memory is not shown.

From this set-up the following things can be deduced from knowing the maximum positive and negative current spikes within 4 time-steps of an input (although this requirement

need not be too stringent if we have a way of recording the maximum current within the ranges in between zeroing the system, which we can do with knowledge of the read pulse clock).

The resulting information from the current is thus:

- 1) if a negative current is recorded in the range -17.5 to $-20nA$: we have had a '1' input into the system
- 2) if a negative current is recorded in the range -5 to $-17.5nA$: we have a carry bit from the operation
- 3) if a negative current is recorded in the range 0 to $-5nA$: we have had a zero in the system (this is redundant information)
- 4) if the maximum positive current is recorded in the range 0 to $+5nA$: the result of the calculation is '0'
- 5) if the maximum positive current is recorded in the range $+5$ to $+9nA$: the result of the calculation is '1'
- 6) if the maximum positive current is recorded in the range $+9$ to $+12.3$: the result is '2' (or '1' for the carry bit, '0' for the summation bit)
- 7) if the maximum positive current is recorded over $12.5nA$: the result is '3' (or '1' for both the carry and summation bit in binary logic).

The output in the negative is purely a result of the input voltages to the system. The positive system includes the 'bounceback', and the summation effect as probed by the read voltage which gives thresholded values of the memristor's state.

With switches, it would be possible to send on the logical result as binary. Region two of the plot encodes the carry bit for the operation, because only if there are two $-M$ spikes (which encode '1') within 3 time-steps of each other we will see a current response in that range. The summation bit is not encoded in as direct a manner, the maximum of the positive currents encodes the numerical sum, and so the summation bit for the value 3 is in a different place to that for the value 1. If we only require knowledge of the carry and summation

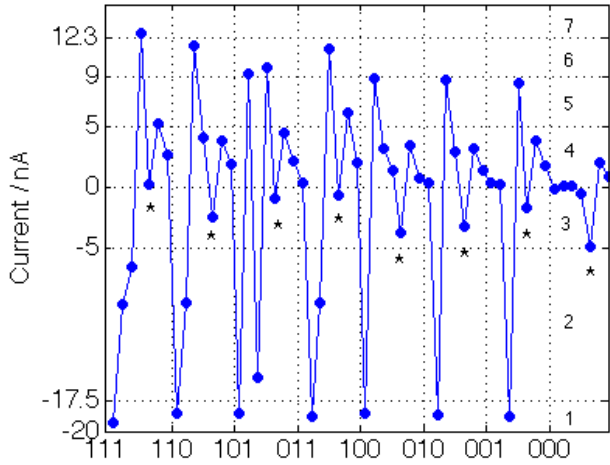


Fig. 4. An attempt at a full adder using mixed logic 2. The first three input bits are the logical inputs, the system has one timestep to respond (t_1) before a read spike is sent in, as marked by an *. The numbers of the ranges correspond to the list.

bit, we can do without the read voltage and corresponding spikes. Changing the values of M and m can tune the effect and might allow us to change the relative values of the output spikes.

V. CONCLUSIONS

In this short paper we have summarised the physical aspects and causes of the spiking interactions observed in a large number of experimental tests and demonstrated some of the interactions via the creation of NOT, AND and Full Adder gates. These gates are not cascable, because the output is a different form to this input, nonetheless, the high degree of functionality of a single memristor suggests that solving this problem will be a worthwhile endeavour.

In terms of logical operational complexity, we are not sure if a Full Adder is the limit for a single memristor. The example shown here takes in 3 bits of information and the output includes: the sum, the value of the carry bit, whether the input includes a 1, whether the input includes a 0 and, from the precise value of the spikes at t_2 , and from a more precise thresholding over the outputs, we can learn where the zero is in the 2 input sequences ([011,101,110]) and where the '1' is in the 1 input sequences ([001,010,100]). These last two points are interesting as it suggests that the memristor Full Adder shown here does not destroy information by the operation, however the gate is not-reversible (as this would require the ability to run time backwards to reverse the physics!). Thus we suggest that with clever design, the memristor can be made to compute more information. As the memristor has to be zeroed, and this takes time, we would want the memristor to do the maximum amount of processing, which suggests that a processor built out of memristors would have a lower clock speed but may compute more bits of information each cycle.

The use of memristor summation approaches in the full adder scheme is similar to how neurons work. For example three '1' inputs received one after the other causes the largest response spike and the only positive t_2 spike, either of these outputs could be linked to a thresholded switch which could release a current or voltage spike and thus act like a leaky integrate and fire neuron. The diminishing returns effect could enforce a refractory period. As neurons work by converting a rate-coded spiking voltage to a current spike at the synapse and then to a voltage spike, all of which can be considered transmission of a logical '1', the memristor with its action whereby input and output are current and voltage, could be ideally suited to neuromorphic computing.

REFERENCES

- [1] L. Chua, V. Sbitnev, and H. Kim, "Hodgkin-huxley axon is made of memristors," *International Journal of Bifurcation and Chaos*, vol. 22, p. 1230011 (48pp), 2012.
- [2] —, "Neurons are poised near the edge of chaos," *International Journal of Bifurcation and Chaos*, vol. 11, p. 1250098 (49pp), 2012.
- [3] S. H. Jo, T. Chang, I. Ebong, B. B. Bhadviya, P. Mazumder, and W. Lu, "Nanoscale memristor device as a synapse in neuromorphic systems," *Nanoleters*, vol. 10, pp. 1297–1301, 2010.
- [4] G. Howard, E. Gale, L. Bull, B. de Lacy Costello, and A. Adamatzky, "Towards evolving spiking networks with memristive synapses," in *Artificial Life (ALIFE), 2011 IEEE Symposium on*, april 2011, pp. 14–21.
- [5] C. T. Themistoklis Prodromakis and L. Chua, "Two centuries of memristors," *Nature Materials*, vol. 11, pp. 478–481, 2012.
- [6] L. O. Chua, "Memristor - the missing circuit element," *IEEE Trans. Circuit Theory*, vol. 18, pp. 507–519, 1971.
- [7] D. B. Strukov, G. S. Snider, D. R. Stewart, and R. S. Williams, "The missing memristor found," *Nature*, vol. 453, pp. 80–83, 2008.
- [8] L. Chua, "Resistance switching memories are memristors," *Applied Physics A: Materials Science & Processing*, pp. 765–782, 2011.
- [9] C. L. S. Grimes and O. Martinsen, "Memristive properties of human sweat ducts," *World Congress on Medical Physics and Biomedical Engineering*, vol. 25/7, pp. 696–698, 2009.
- [10] E. Gale, B. de Lacy Costello, and A. Adamatzky, "Observation, characterization and modeling of memristor current spikes," *Appl. Math. Inf. Sci.*, vol. 7, pp. 1395–1403, 4 July 2013.
- [11] —, "Observations of bursting spike patterns in simple memristor circuits," in *Submitted*.
- [12] —, "Boolean logic gates from a single memristor via low-level sequential logic," in *Submitted*.
- [13] E. Gale, D. Pearson, S. Kitson, A. Adamatzky, and B. de Lacy Costello, "Aluminium electrodes effect the operation of titanium oxide sol-gel memristors," *arXiv:1106.6293v1*, 2011.
- [14] E. Gale, R. Mayne, A. Adamatzky, and B. de Lacy Costello, "Drop-coated titanium dioxide memristors," *Materials Chemistry and Physics*, vol. 143, pp. 524–529, January 2014.
- [15] J. Borghetti, G. D. Snider, P. J. Kuekes, J. J. Yang, D. R. Stewart, and R. S. Williams, "'memristive' switches enable 'stateful' logic operations via material implication," *Nature*, vol. 464, pp. 873–876, 2010.