2010 12th International Workshop on Cellular Nanoscale
Networks and their Applications (CNNA)

Arithmetic Operations within Memristor-Based Analog Memory

Mika Laiho, Eero Lehtonen
Microelectronics Laboratory, University of Turku
Joukahaisenkatu 3-5, 20014 Turku, Finland
Email: {mlaiho, elleht}@utu.fi

***Abstract*—This paper describes how memristors could be used as an analog memory and computing elements. The key idea is to apply comparison and programming phases cyclically so that the memristor can be programmed to a given conductance level at a fixed voltage. It is further described how the cyclical programming could be used in computing. A configuration needed to copy the sum of conductances of two memristors into a third one is described. It is further shown how the devices could be configured so that addition and subtraction of positive and negative analog conductances could be performed. The presented memory structure requires a memristor model with a nonlinear programming sensitivity (programming threshold) for proper programming selectivity. A model of such a memristor is shown and key simulations are presented.**

I. Introduction

The existence of fourth passive circuit element, memristor was formulated by Chua in 1971 [1], but it did not draw much attention since a physical implementation was unavailable. After HP Labs published their experimental results of TiO_2-based memristors [2], there has been a strong interest in using memristors for different nonvolatile memory applications. Mostly, memristors have been proposed to act as digital memory elements, storing one bit of information, i.e., programming the memristor to either conducting (ON) or unconducting (OFF) states. However, it has been experimentally shown that memristors can be programmed to intermediate conducting states also [3]. It is not yet known how accurate the programming can be since a finite number of mobile oxygen vacancies participate in the channel forming; this could lead to a discrete spectrum of possible conductance values for the memristor. Therefore, this paper does not deal with the issue of accuracy, but rather assumes that a sufficient accuracy is available for a specific application.

When a CMOS process is complemented with memristors by processing crossbar memristor arrays on top of the CMOS stack, the resulting CMOS-memristor hybrid (CMOL) technology offers exciting possibilities for, e.g., realization of artificial neural networks [4]. The same applies to cellular neural/nanoscale networks (CNN) [5], arrays of locally connected processing cells that are capable of powerful spatial-temporal processing. A CNN complemented with cell level additions, e.g., analog memory and arithmetic processing capabilities is referred to as CNN universal machine (CNNUM). The analog memories and analog arithmetic processing capabilities require a lot of silicon area, so their realization with memristors would significantly compress the overall cell area. Yet another processor array that is based on local interaction, namely the analog microprocessor (AμP) [6], could benefit from the use of memristors. In the AμP, neighborhood interaction and local arithmetic are all performed using analog current memories. For example, addition of the contents of two memories is carried out by reading out the currents of these simultaneously, and writing the sum current to a third one.

In this paper, a memristor-based analog memory is presented. First, an appropriate memristor model that has a nonlinear programming threshold is described. Then, it is shown how addition and subtraction can be performed in analog domain using the memristors.

II. Memristor Model for Analog Memory

For memory applications, it is required that a read-operation does not destroy the data, or, if it does, the data can at least be somehow restored. The memristor reported in [2] gets programmed linearly, as charge is passed through the device. Therefore, it is difficult to read the resistive memory without affecting the contents. Some AC readout means could provide a solution, but here a different approach is taken. The resistance of the memristor reported in [3] exhibits a nonlinear dependence on the voltage over the memristor. A strong nonlinearity results effectively in a programming threshold: the device can be operated either in programming or readout mode depending on the voltage over the device. In this paper a SPICE model introduced in [7], based on the experimental data and model of [3] is used. In the model of [7], the current through the memristor is

$$I = (b + W^n)\beta \sinh(\alpha V) + \chi(\exp(\gamma V) - 1). \quad (1)$$

Here W is the state variable of the memristor [2] and the constants α, β, χ, γ and b depend on the physical properties of the component. The time derivative of the state variable W is modeled as

$$\frac{\mathrm{d}W}{\mathrm{d}t} = a \cdot f(W) \cdot V^q, \quad (2)$$

where a is a constant, $f : [0, 1] \rightarrow \mathbb{R}$ is a *window function* [2]

$$f(W) = 1 - (2x - 1)^{2p} \quad (3)$$

where p is a positive integer. The state derivative depends on the q'th power of V, making a proper programming threshold for large values of q. In the simulations of this paper a q

978-1-4244-6678-8/10/$26.00 ©2010 IEEE

of 13 was used. Figure 1 shows Eldo simulations (Eldo is a commercial variant of SPICE) of the memristor model. The simulations are carried out with slightly larger β, χ and a compared to [7] in order to increase the current level and slow down the programming. The left subfigure shows voltage over the memristor, along with the memristor state variable W as a function of time. It is obvious that small voltages over the memristor do not affect W which demonstrates the existence of a proper programming threshold (about 1.8V). The subfigure on the right shows the current-voltage characteristics of a memristor showing a proper signature of a memristive device. It should be emphasized that it is not aiming to be an accurate physical model of [3] but rather the aim is at reproducing the general characteristics/behavior of a memristor. Also, it should be noted that here the state variable W is hard limited between 0.05 and 0.95. This avoids the problems in the extremes of W described in [8].

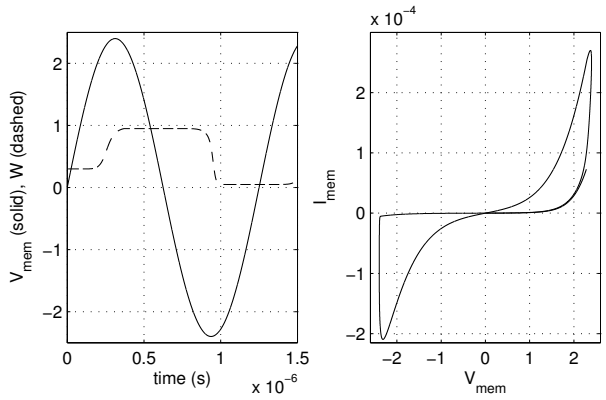

Figure 1. Memristor characteristics simulated using SPICE model.

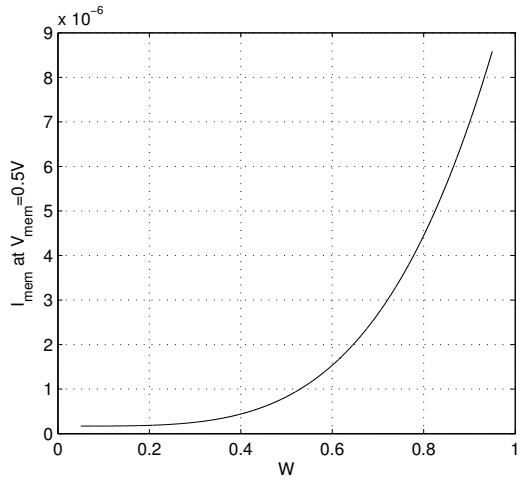

Figure 2. Simulated memristor current I_{mem} with varying W at a constant voltage over memristor V_{mem}

III. Memristor Memory

The memristor I/V curve of Figure 1 reveals that the current through the memristor is highly nonlinear (combination of hyperbolic sine and exponential characteristics). Therefore, readout voltage over the memristor should always be the same when using the memristor as an analog memory. A further consequence of the nonlinearity is that it is not straightforward to control the instant when programming of the memristor has reached the target value. The situation is analogous to using programming floating gate- based analog memory elements that rely on successive application of programming and monitoring phases [9]. In the monitoring phase the current through the memristor is read at a constant voltage. Figure 2 shows the simulated memristor current I_{mem} with varying W at a constant voltage over memristor V_{mem} of 0.5V.

Figure 3 shows the schematic of the memristor- based analog memory/arithmetic computing unit. It shows six memristors, controlled with node voltages V_1- V_6. These nodes can be set to high impedance state, or be programmed to voltages VDD_P, VDD_R, VSS_R and VSS_P. These voltages are programmed to 2, 1.5, 0.5 or 0 V, respectively, while the analog ground level GND is at 1V. The schematic also shows a number of switches and resistor R_0 that can be used to write a reference resistance value to a memristor, or can be used to read out a memristance value by converting the schematic to an inverting operational amplifier.

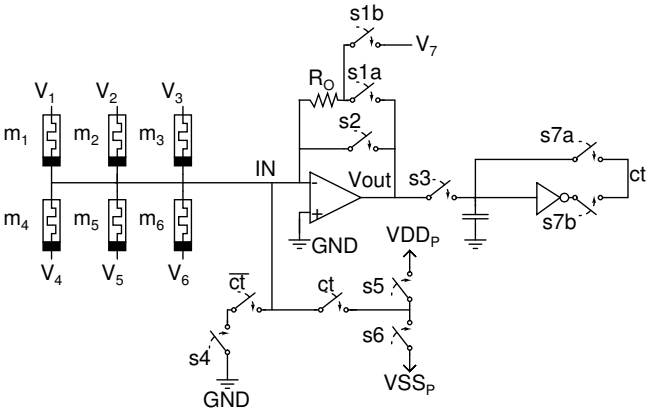

Figure 3. Memristor-based analog memory/computing unit.

Figure 4 shows a simulation in which the conductance value of R_O is copied to memristor m_1. In this operation the control signals $s1b$ and $s7a$ ON. Prior to programming, m_1 has been programmed to a low conductance state. The programming proceeds so that in the monitoring phase $S4$ is ON, whereas $S6$ is OFF. Once the operational amplifier output V_{out} is stable, this voltage is sampled with $S3$ to the capacitor. The capacitor voltage ct controls whether programming occurs (node IN is pulled to VDD_P or the targeted level has been reached (node IN is pulled to GND through switch $S4$. In the simulation, V_7=VSS_R. Since R_O is 200kΩ ($5\mu S$) and programming stops when V_{in} equals GND, both R_O and m_1 end up with the same conductance value. The top subfigure of Figure 4 shows how voltage over m_1 alternates between programming and monitoring phases. The memristor voltage ends at 0.5V. Each programming cycle increases the voltage

at V_{in} as m_1 becomes more conductive. The second subfigure shows that current through m_1 (and R_O) ends up with a value of $2.5\mu A$, corresponding to $5\mu S$ since memristor voltage is $0.5V$. Also shown in the figure is ct that goes low when the targeted value has been reached. The lowest subfigure of Figure 4 demonstrates that the W of the memristor ends up at a value of approximately 0.7.

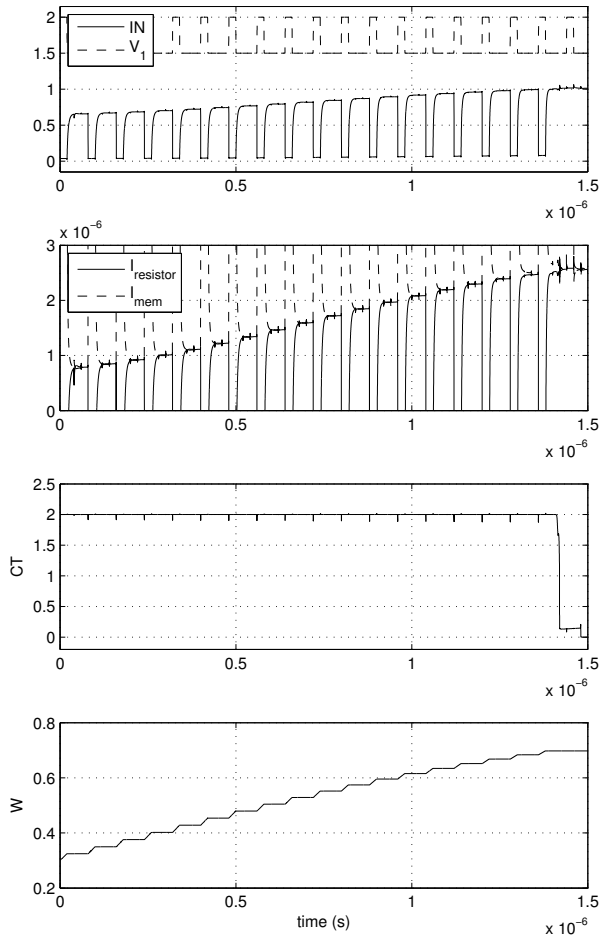

Figure 4. Memristor characteristics simulated using SPICE model.

It should be noted that when the programming ends, (ct goes LO) the voltages over all memristors are constant if $VDD_R - GND = GND - VSS_R$. Summing currents of resistive components with a fixed voltage is the same as summing their conductances. The operation of the circuit during summing of two conductances can be observed from Figure 3. In this example the sum of conductances of m_4 and m_5 is written to m_1. It is assumed that memristor m_1 is initially programmed to low conductance (OFF) state. Figure 3a) shows the monitoring phase. If $g_{m4} + g_{m5} > g_{m1}$ (g stands for the conductance), control voltage ct will be HI. The voltage stored at ct determines whether programming will take place in the subsequent programming phase. If ct is HI, $VDD_P - VSS_P = 2V$ will appear over the memristor. A sequence of monitoring and programming cycles assures that $g_{m1} \geq g_{m4} + g_{m5}$. The amount of programming during one cycle (step size) can be reduced by shortening the duration of a programming cycle, or lowering VDD_P. If the step size is small enough, $g_{m1} \approx g_{m4} + g_{m5}$.

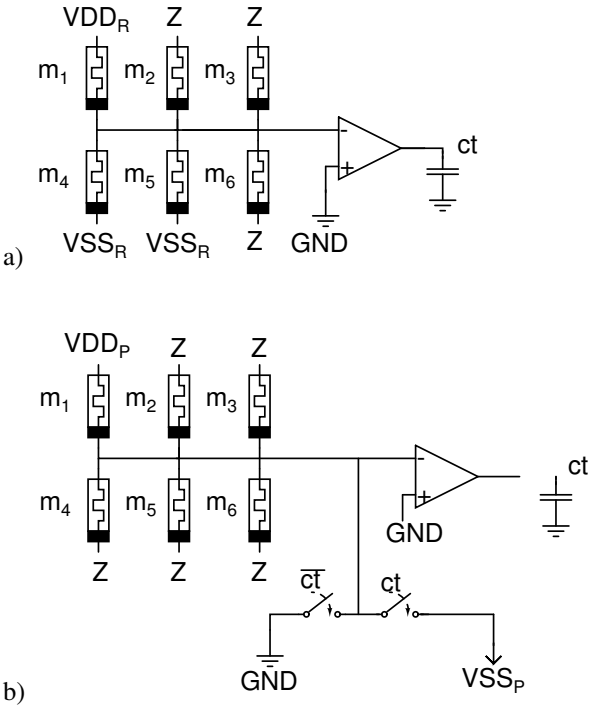

Figure 5. Processing example: sum of conductances. a) Monitoring phase. b) Programming phase.

If one wants to be able to sum both positive and negative numbers, the memristor memories can be configured as with the analog microprocessor [6]. In this case each analog memory is composed of two memristors as shown in Figure 6a). The upper memristor of a memory is always programmed to the middle of the conductance range, whereas the lower conductance can be programmed between low and high conductance states. Figure 6b) shows a copy operation between memories. The conductance values of the memory on the left are 0.5 and 0.3, representing the number -0.2. In the copy operation, this values is inverted analogously to the current memories the analog microprocessor [6].

Figure 7 shows an addition operation where the sum of memory 1 and memory 2 is written to memory 3. Note that the sum value is inverted as in Figure 6.

In the analog microprocessor there is a binary weighted array of current mirrors that can be used to perform multiplication and division operations. Since memristors are two-terminal devices, they cannot be used as controllable sources. Conductance values could be multiplied by integer numbers by repeatedly summing conductance values. However, mul-

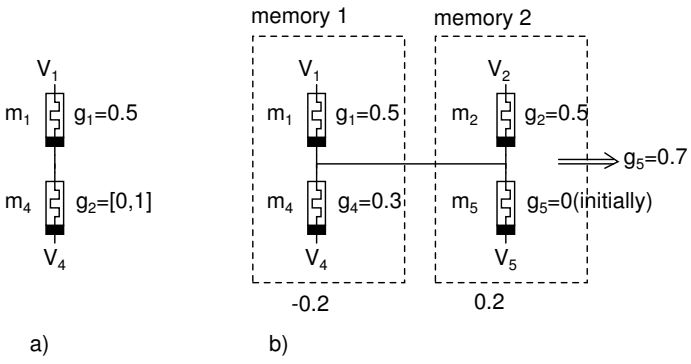

Figure 6. a) memristor-based memory that can store both positive and negative values. b) inversion (multiplication by -1) takes place when contents of memristor memory is copied to another.

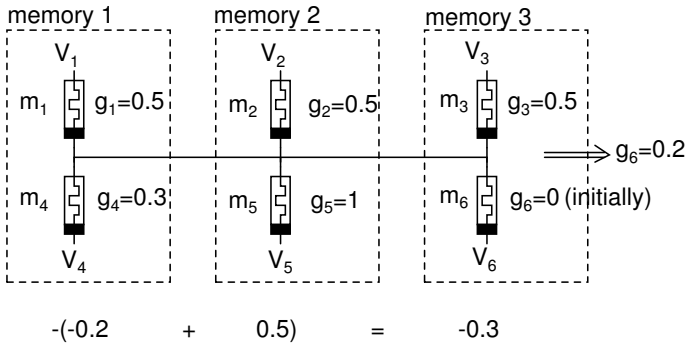

Figure 7. Addition of the contents of two memristor memories and the storage of the results to a third one. Note that the result is inverted as in the copy operation of figure 6.

tiplication and division by noninteger numbers is a topic of future research.

IV. Conclusions

In this paper we showed how memristors could be used as analog memories and used for computing. A cyclical programming method was proposed that automatically stops the programming when the correct conductance value has been reached. A two-memristor configuration was proposed to be used as a memory element so that addition operations of both positive and negative numbers could be performed. Key simulations of the circuit were carried out using Eldo circuit simulator. The memristor model used in the simulations has a proper programming threshold that is needed in selective programming of the memristive devices.

Acknowledgments

This work was funded by the Academy of Finland (131295), Nokia foundation, and the GETA graduate school.

References

[1] L. O. Chua, “Memristor - The Missing Circuit Element”, IEEE Transactions on Circuit Theory, vol. 18, no. 5, pp. 507-519, 1971.

[2] D. Strukov, et al., “The Missing Memristor Found”, Nature, vlo. 453, pp. 80-83, May 2008.

[3] J. Yang, et al., “Memristive Switching mechamism for metal/oxide/metal nanodevices”, Nature Nanotechnology, Vol. 3, July 2008.

[4] Ö. Turel, et al., “Neuromorphic Architectures for Nanoelectronic Circuits”, International Journal of Circuit Theory and Applications”, Vol., 32, pp. 277-302, 2004.

[5] L.O.Chua, L.Yang, “Cellular Neural Networks: Theory”, *IEEE Transactions on Circuits and Systems-I*, vol.35, no.10, pp.1257–1272, Oct. 1988.

[6] P.Dudek and P.J.Hicks, "A CMOS General-Purpose Sampled-Data Analogue Microprocessor", IEEE International Symposium on Circuits and Systems, ISCAS 2000, Geneva, Switzerland, vol.II, pp.417-420, May 2000.

[7] E. Lehtonen, M. Laiho, “CNN Using Memristors for Neighborhood Connections”, submitted to IEEE Cellular Nanoscale Networks and their Applications, 2010.

[8] Z. Biolek, D. Biolek, V. Biolkova, “SPICE Model of Memristor with Nonlinear Dopant Drift”, 2009.

[9] A. Bandyopadhyay, G.J. Serrano, P. Hasler “Adaptive Algorithm Using Hot-Electron Injection for Programming Analog Computational Memory Elements Within 0.2% of Accuracy Over 3.5 Decades”, IEEE Journal of Solid-State Circuits, Vol. 41, Issue 9, pp. 2107 - 2114, Sept. 2006.