SPICE Modeling of Memcomputing Logic Gates

Yuriy V. PERSHIN

Dept. of Physics and Astronomy, University of South Carolina, Columbia, SC 29208 USA

pershin@physics.sc.edu

Submitted July 29, 2023 / Accepted October 17, 2023 / Online first November 6, 2023

Abstract. *Memcomputing logic gates generalize the traditional Boolean logic gates for operation in the reverse direction. According to the literature, this functionality enables efficient solution of computationally intensive problems, including factorization and NP-complete problems. To approach the deployment of memcomputing gates in hardware, this paper introduces SPICE models of memcomputing logic gates following their original definition. Using these models, we demonstrate the behavior of single gates as well as small self-organizing circuits. We have also corrected some inconsistencies in the prior literature. Notably, the correct schematics of the dynamic correction module is reported here for the first time. Our work makes memcomputing more accessible to those interested in this emerging computing technology.*

Keywords

Memristors, SPICE, nonlinear dynamical systems, computing technology

1. Introduction

Digital memcomputers are an emerging class of unconventional computing systems developed to efficiently solve factorization and combinatorial optimization problems [1], [2]. Fundamentally, these are complex dynamical systems with deterministic continuous dynamics whose phase space contains a fixed point attractor corresponding to the problem solution (or multiple attractors if several solutions are possible). According to Traversa and Di Ventra, when implemented in hardware, digital memcomputing machines offer a polynomial-time solution to factorization and NP-complete problems [1]. Moreover, it has been argued that the dynamics of digital memcomputing machines (with solutions) is deterministic, non-chaotic, without periodic orbits [3], [4], and topologically robust against perturbations and noise [5], [6]. If a solution exists, it is certain that it will be discovered. Otherwise, there is no equilibrium. In such cases, some or all of the gates will continually change in an effort to self-organize into the logically consistent state of the circuit. However, such a state does not exist in problems without a solution.

So far, the research on digital memcomputing has been substantially focused on software simulations of ordinary differential equations representing the circuit dynamics. A recent benchmark [7] indicates that the memcomputing solution for a specific class of difficult problems is characterized by an exponent similar to that found for other solvers (such as Toshiba simulated bifurcation machines [8] and Fujitsu digital annealers [9]). Further progress could be made if computing was implemented on the hardware. For more information on competitive computing approaches, we refer the interested reader to [7] and the references therein.

Three designs of memcomputing logic gates are available in the literature. The most complex is the original design [1] (Design I) that is presented in Fig. 1. According to Fig. 1, self-organizing AND, OR or XOR can be built using 12 memristive elements, 15 voltage-controlled voltage generators, and several resistors. Moreover, some auxiliary circuitry is required to ensure that the final states of these gates (operating in the continuous or analog domain) are binary. Bearden et al. [10] introduced a simplified design of self-organizing AND and OR (Design II) [10] in which each gate requires 5 memristive elements, 6 voltage generators and few resistors. In principle, the simplified gates can perform the same tasks as the original ones. A study shows that Design II gates can be built using physical memristive devices [14]. In the third design (Design III), the gates are defined by a set of differential equations [15]. In fact, Design III is a variation of analogSAT [16], [17]. Recently, Design III calculations were implemented on an FPGA board [18].

The purpose of this work is to develop SPICE models of Design I self-organizing logic gates [1]. During the last decade or so, the SPICE modeling of adaptive circuit elements (known as memrisitive, memcapacitive, and meminductive systems [11,12,19]) has become increasingly important and resulted in various SPICE models of deterministic memelements (see, for instance, [20–28]). A notable recent development is the simulation of probabilistic memristive devices in SPICE [29]. SPICE is a general-purpose circuit simulation program [30], [31]. In SPICE, the circuit can be built graphically and then numerically simulated using predefined or user-defined models of individual circuit components. Such user-defined models of Design I self-organizing logic gates [1] are formulated in this paper.

DOI: 10.13164/re.2023.0542

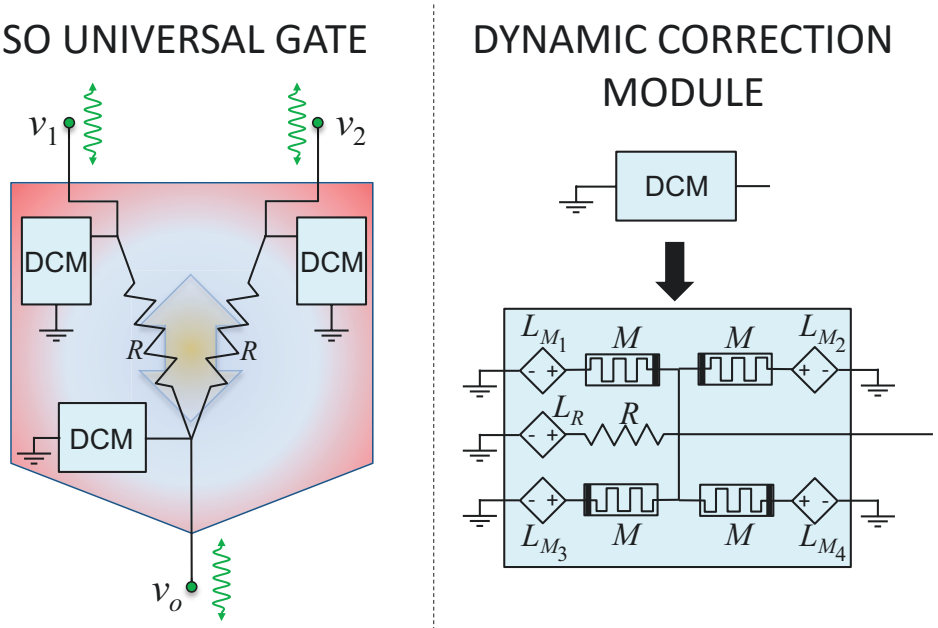


Fig. 1. Left panel: the universal self-organizing logic gate is composed of three dynamic correction modules (DCMs) and two resistors. Right panel: internal structure of the dynamic correction module (incorrect). Here, the resistive memories M [11–13] have minimum and maximum resistances R_{on} and R_{off}, respectively, and the resistance of the resistors is $R = R_{\text{off}}$. L_{M_j}-s and L_R are voltage-controlled voltage generators. As we explain in the text, in the right panel, the polarity of all memristive elements must be reversed. Reprinted from [1], with the permission of AIP Publishing.

In this work, we used LTSpice XVII (Analog Devices) as a simulation tool. Our SPICE models may need minor adjustments to be used on other SPICE simulators (e.g., PSPICE, Ngspice).

The paper is organized as follows. We start with the introduction of Design I self-organizing logic gates (Sec. 2) that is followed by a presentation of their SPICE models (Sec. 3). In Sec. 4, we give examples of SPICE simulations of individual gates and circuits thereof. Section 5 concludes the paper. Complete listings of LTspice codes are given in Appendix C.

2. Self-Organizing Gates and Circuits

Self-organizing logic gates generalize traditional logic gates for operation in the reverse direction [1], [2]. In these gates, each terminal serves the double function of input and output. Although self-organizing gates operate in analog mode, the auxiliary circuitry (voltage-controlled differential current generators presented below) and the gate design ensure that the final states are binary. In what follows, Boolean 1 and 0 are represented by $v_c = 1$ V and $-v_c = -1$ V voltage levels, respectively.

The logic behind the construction of self-organizing gates can be partially captured from the following excerpt from [1]: "if the gate is connected to a network and the gate configuration is correct, no current flows from any terminal (the gate is in stable equilibrium). Otherwise, a current of the order of v_c/R_{on} flows with the sign opposite to the sign of the voltage at the terminal." Below, this property of self-organizing gates is used to demonstrate that the polarity of the memristive elements in Fig. 1 circuit must be reversed for the correct gate operation.

Transformation of traditional Boolean logic circuits into Design I self-organizing logic circuits [1] involves the following steps:

- Replacing the traditional logic gates with self-organizing gates of the same type.
- Representing the external input signals by constant-value voltage sources.
- Adding auxiliary circuitry: A voltage-controlled differential current generator (VCDCG) is added to each node, but not to the nodes that are used for input signals. By node, we mean either the point of connection of two or more gate terminals or an unconnected gate terminal.
- External input signals are applied at the initial moment of time and stay constant over time. The end of dynamics indicates that a solution is found and can be read. The infinite dynamics implies the absence of a solution.

For the sake of completeness, we next provide the minimal description of the circuit components in the Design I self-organizing logic circuits that is required for their implementation in SPICE. These definitions were extracted from [1, 2, 32].

2.1 Voltage-Controlled Voltage Generators

Consider the structure of the universal self-organizing gate shown in Fig. 1. Its ultimate functionality (e.g., self-organizing AND, OR, or XOR) is defined by the equations that govern voltage-controlled voltage generators (VCVGs) $L_{M_1} - L_{M_4}$ and L_R. According to [1], the voltage across VCVG is a linear function of the gate voltages

$$v_{\mathrm{VCVG}} = a_1 v_1 + a_2 v_2 + a_0 v_0 + dc \tag{1}$$

where v_1, v_2, and v_0 are the gate voltages, and a_1, a_2, a_0, and dc are the constants. For convenience, these constants are specified in Tab. A1 (Appendix A).

2.2 Memristive Elements

Having defined the voltage-controlled voltage generators, next we consider the memristive elements M in Fig. 1. The response of the memristive system j is described by Ohm's law

$$v_{M_j}(t) = M(x_j) i_{M_j}(t) \tag{2}$$

where v_{M_j} is the voltage (defined with respect to the thick-bar terminal of the circuit symbol of M), i_{M_j} is the current,

$$M(x_j) = (R_{\mathrm{off}} - R_{\mathrm{on}})\, x_j + R_{\mathrm{on}} \tag{3}$$

is the state-dependent resistance (memristance), $x_j \in [0, 1]$ is the internal state variable [11], R_{on} and R_{off} are the on- and off-state resistances.

The dynamics of x_j follows the ordinary differential equation:

$$\frac{\mathrm{d}x_j}{\mathrm{d}t} = -\alpha h(x_j, v_{M_j}) \left[(R_{\mathrm{off}} - R_{\mathrm{on}})\, x_j + R_{\mathrm{on}}\right]^{-1} v_{M_j} \tag{4}$$

where α is the constant and the function $h(x, v_M)$ influences the dynamics of the internal state. While many choices for $h(x, v_M)$ are available, following [1], we use

$$h(x, v_M) = \theta(x)\,\theta(v_M) + \theta(1 - x)\,\theta(-v_M) \tag{5}$$

where $\theta(\dots)$ is the unit step function. According to (4) and (5), x_j decreases down to $x_j = 0$ at positive voltages. At negative voltages, x_j increases to $x_j = 1$. It is not difficult to recognize that (2), (4), and (5) correspond to the ideal memristor model [33]. The limitations of this model are well known [19]. Note that Table A2 (Appendix A) lists the values of parameters used in the prior simulations [1]. Equation (5) corresponds to $V_t = 0$ and $k = \infty$. For definition of V_t and k, see [1].

Moreover, a parasitic device capacitance is taken into account using a constant-value capacitor of capacitance C connected in parallel to every memristive element. These capacitors are not shown in Fig. 1 explicitly but taken into account in the model.

2.3 Voltage-Controlled Differential Current Generators

Finally, we introduce equations describing voltage-controlled differential current generators. These generators are second-order dynamical systems whose evolution follows [1], [32]:

$$\frac{\mathrm{d}i_{\mathrm{DCG},j}}{\mathrm{d}t} = \theta\left(s_j - \frac{1}{2}\right) f_{\mathrm{DCG}}(v_{\mathrm{DCG},j}) - \gamma\theta\left(\frac{1}{2} - s_j\right) i_{\mathrm{DCG},j}, \tag{6}$$

$$\frac{\mathrm{d}s_j}{\mathrm{d}t} = f_s\left(\mathbf{i}_{\mathrm{DCG}}, s_j\right). \tag{7}$$

Here, γ is the constant, $v_{\mathrm{DCG},j}$ is the voltage at the node that VCDCG is connected to, $i_{\mathrm{DCG},j}$ is the current, $\mathbf{i}_{\mathrm{DCG}}$ is the vector of the currents of all VCDCGs, and s_j is the second state variable. Note that the above equation corresponds to $\delta s = 0$ in Tab. A2 (Appendix A). For the definition of δs, see [1]. Our specific realization of $f_{\mathrm{DCG}}(x)$ in (6) is based on arc tangent functions (one of the suggested realizations of $f_{\mathrm{DCG}}(x)$ [1]). Specifically, we use

$$f_{\mathrm{DCG}}(x) = \frac{2q}{\pi}\left(\arctan\left[\frac{m_1\pi}{2q}(x + v_{\mathrm{c}})\right] + \arctan\left[\frac{m_0\pi}{2q}x\right] + \arctan\left[\frac{m_1\pi}{2q}(x - v_{\mathrm{c}})\right]\right) \tag{8}$$

where q, m_0, m_1, and v_{c} are the constants. Equation (8) is illustrated in Fig. 2(a).

In (7), the function $f_s\left(\mathbf{i}_{\mathrm{DCG}}, s_j\right)$ is defined as

$$f_s\left(\mathbf{i}_{\mathrm{DCG}}, s_j\right) = -k_s s_j (s_j - 1)(2s_j - 1) - k_i\left(1 - \prod_p \theta\left(i_{\min}^2 - i_{\mathrm{DCG},p}^2\right) - \prod_p \theta\left(i_{\max}^2 - i_{\mathrm{DCG},p}^2\right)\right). \tag{9}$$

Here, k_s, k_i, $i_{\min}$, and $i_{\max}$ are the positive constants ($i_{\min} < i_{\max}$). Equation (9) corresponds to $\delta i = 0$ in Tab. A2 (Appendix A). For the definition of δi, see [1].

Importantly, unlike [1], [32] we use the minus sign in front of the k_i term in (9). The minus sign is required to implement the following anticipated purpose of s_j-s: the reset of all $|i_{\mathrm{DCG},j}|$ to below $i_{\min}$ as soon as at least one of $|i_{\mathrm{DCG},j}|$ exceeds $i_{\max}$.

Equation (9) is illustrated in Fig. 2(b). Figure 2(b) (left panel) shows that when the absolute value of all currents is less than $i_{\min}$, the function $f_s\left(\mathbf{i}_{\mathrm{DCG}}, s_j\right)$ has a single zero at $s > 1$, which is a stable fixed point of (7). When the absolute value of at least one $i_{\mathrm{DCG},p}$ is larger than $i_{\max}$, the stable fixed point is located at $s < 0$, see Fig. 2(c) (right panel). In the intermediate case, there are two stable fixed points and one unstable fixed point (Fig. 2(b) (middle panel)). Therefore, when the absolute value of one of the currents exceeds $i_{\max}$, all variables s_j – which are described by identical equations – start drifting toward the negative stable fixed point, causing the relaxation of VCDCG currents (through the last term in (6)). The normal response of VCDCGs (due to the first term ino. (6)) is restored later, after the condition $|i_{\mathrm{DCG},j}| < i_{\min}$ for all j is satisfied.

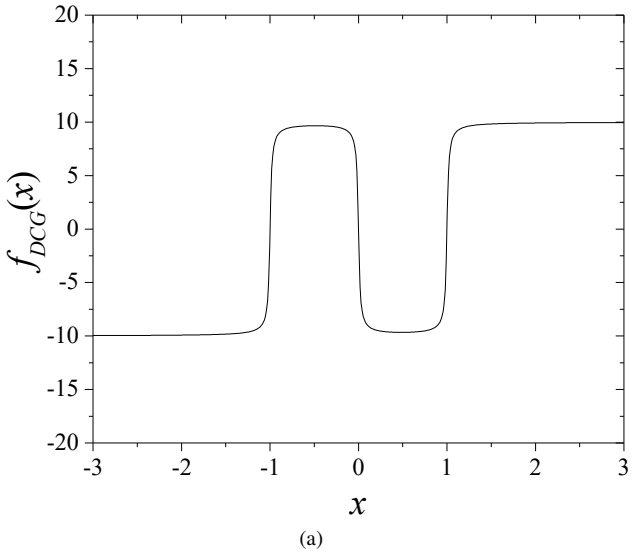


Fig. 2. Functions (a) $f_{\mathrm{DCG}}(x)$ and (b) $f_s\left(\mathbf{i}_{\mathrm{DCG}}, s_j\right)$ defined by (8) and (9), respectively. These graphs were obtained using the parameters values from Tab. A2 (Appendix A).

3. SPICE Models

3.1 Basic Details

Our SPICE models are based on the parameters' values from [1] (Tab. A2, Appendix A) with the assumption that these values are given in the SI units. An issue is that the small resistances lead to large currents (of the order of amperes) and some other parameters lead to slow dynamics (on the scale of seconds). In particular, the on-and off-state resistances, $R_{\mathrm{on}} = 0.01\,\Omega$ and $R_{\mathrm{off}} = 1\,\Omega$, are much smaller than the on- and off-state resistances in experimental devices that typically range from kiloohms to megaohms. In our LTspice models, we have introduced two scaling factors, z_I and z_t. These factors are used to rescale the currents and time in typical ranges for electronics. For consistency, the same values of z_I and z_t must be utilized in all LTspice models (Appendix C). The results reported here were obtained using $z_I = 10^5$ and $z_t = 10^3$.

Appendix C contains LTspice models of self-organizing AND, OR, and XOR, memristive elements, and voltage-controlled differential current generators (Listings C1–C5). These models were formulated following the common practices in SPICE modeling [22]. For example, to integrate the differential equations (4), (6), and (7), we use capacitors that are charged or discharged with voltage-controlled current sources representing the right-hand sides of these equations, etc.

Figure 3 shows the current-voltage curves for the memristive element. To obtain these curves, we used the SPICE model from Listing C4. According to Fig. 3, the memristance decreases at $V_M > 0$ and increases at $V_M < 0$. The frequency dependence of the current-voltage curves in Fig. 3 is typical for memristive systems [11], [12].

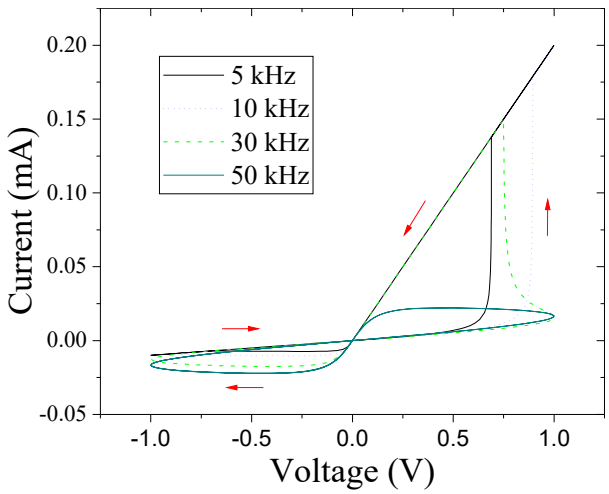


Fig. 3. Current-voltage curves of the memristive element subjected to a sinusoidal voltage. These curves were obtained using the SPICE model in Lst. C4 (Appendix C).

We have found that the polarity of the memristive elements in Fig. 1 is incorrect. To understand this, one can evaluate the total terminal current in a gate terminal assuming a logically consistent state. As we have mentioned above, in this case, the total terminal current must be zero (see the second paragraph in Sec. 2.1). For example, a simple calculation shows that the current at terminal 1 of AND at $V_1 = V_2 = V_o = -1$ V is $I_1 = 2/M_1 - 2/R_{\rm off}$ and the voltage across M_1 in Fig. 1 circuit is +2 V. As positive voltages drive M_1 into $R_{\rm on}$, $I_1 = 2/R_{\rm on} - 2/R_{\rm off} > 0$. Thus, to satisfy $I_1 = 0$, the polarity of M_1 must be reversed.

As all variables s_j are described by the same Equation (7), we represent these variables by a single variable $s \equiv s_j$ for all j. In SPICE, we have defined an s-block as the component that integrates (7) (see the second model in Listing C5 and Fig. B1 in Appendix B). The s-block has 8 voltage inputs for the voltage signals encoding $i_{{\rm DCD},j}$-s and a single output, which is s. The output of s-block must be connected to all VCDCGs (to terminals 4). The inputs of the s-block are taken from terminals 3 of the VCDCGs. The SPICE model of the s-block in Listing C5 can be directly used with up to 8 VCDCGs in the circuit and can be easily expanded to a larger number of VCDCGs.

3.2 Important Implementation Notes

1. The correct version of Design I self-organizing gates involves two resistors (Fig. 1, left). In [2], it is shown without resistors. In [32], it is presented with memristors instead of resistors.

2. The polarity of memristors in Fig. 1 has been reversed, see Fig. 4.

3. We emphasize that unlike [1], [32], we use the minus sign in front of k_i in (9).

4. The values $k_i = 10^{-7}$ and $k_s = 10^{-7}$ from [1], [32] are too small. To enable the correct operation of the reset feature in VCDCGs, we use $k_i = k_s = 2{\rm E}3$.

5. Memristive elements subjected to zero voltage and associated voltage sources have not been included in the models of AND and OR (e.g., L_{M_2} and L_{M_4} and associated memristive elements in the DCM of terminal 1 of AND).

6. All s_j-s are implemented using a single variable s.

7. To suppress high voltage spikes[1] and improve the convergence, we have increased $R_{\rm on}$ to 0.05, decreased $i_{\rm max}$ to 10, and added a capacitor in each VCDCG (C1 in Listing C5).

8. To enable the random initial states of memristive elements, the option "Use the clock to reseed the MC generator" must be checked in LTSpice XVII. The transient analysis was performed using the option *uic*.

9. For reproducibility of our results, the initial states of the memristive elements are chosen from a flat random distribution between 0.18 and 0.22. All other initial values are selected deterministically.

4. Simulation Examples

4.1 Single Gates

This subsection exemplifies the behavior of self-organizing gates using the OR gate as an example. First, we consider the traditional (direct) operation, wherein the voltage signals are applied to the traditional inputs of the gate. Second, we explore the reverse operation (not available with the usual OR).

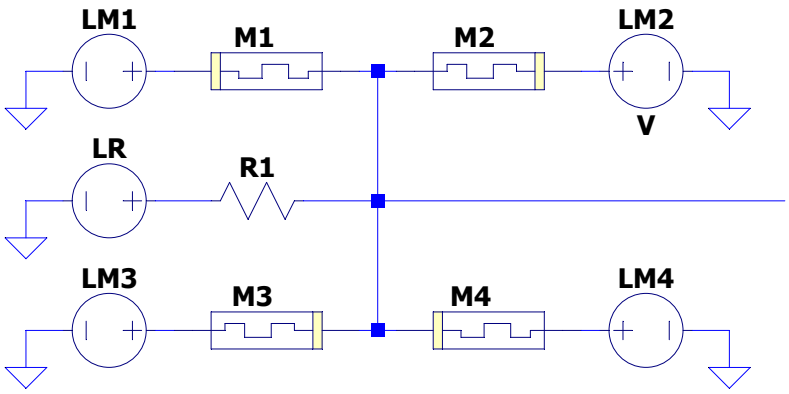


Fig. 4. Correct schematics of the dynamic correction module.

[1] In the circuit based on original parameters, spikes can be of quite extreme magnitude (e.g., several hundred thousand volts). These spikes have been associated with instantons, see [2].

Figure 5(a) shows the simulated circuit for the direct operation of the self-organizing OR. In this circuit, the input signals are applied using pulsed voltage sources connected to terminals 1 and 2 of U1. The output terminal of U1 (not connected to any voltage source) is driven by a VCDCG (see the circuit transformation rules in Sec. 2). Figure 5(b) shows the gate response. Clearly, except for short transients, the gate reproduces the truth table of OR. In Fig. 5(b), the curve V_{VCDCG} represents the current in VCDCG U2 (without accounting for the current of C1). We note that this current fluctuates at about zero.

Next, consider the reverse operation of the self-organizing OR. Figure 6(a) and (b) show two slightly different circuits used in our simulations. The difference is that in Fig. 6(a) we use two terminals (terminals 2 and o) as input and one terminal (1) as output, while in Fig. 6(b) there is a single input terminal (o) and two output terminals (1 and 2). As before, we use VCDCGs to ensure binary states at the output terminals.

Figure 6(c) shows the gate voltages for the circuit in Fig. 6(a). In this presented realization of circuit dynamics, after a transient, the voltage at the first terminal, $V_1(t)$, becomes equal to -1 V. When $V_o = 1$ V, the truth table of OR is satisfied. In the opposite case, the gate shows a reasonable behavior as it chooses $V_1 = -1$ V over $V_1 = 1$ V deterministically. In some other runs, instead of $V_1(t) = -1$ V, the voltage at terminal 1, after a short transient, repeated the applied voltage $V_o(t)$. In this case, again, the truth table of OR is satisfied whenever $V_o(t) = 1$ V.

Finally, consider the response of the self-organizing OR in Fig. 6(b). Using a flat random distribution of initial states of memristive elements from 0 to 1, in each run we observed one of the following general responses: (i) $V_1(t) = V_2(t) = V_o(t)$ (as in Fig. 6(d)), $V_1(t) = -1$ V, $V_2(t) = V_o(t)$, or (iii) $V_1(t) = V_o(t)$, $V_2(t) = -1$ V. Clearly, all of these cases are consistent with the truth table of the OR.

Overall, we conclude that the self-organizing OR correctly reproduces the truth table of OR (whenever possible) regardless of the role of each terminal as input or output. The response may be different in different runs (but always correct). We have verified that the same is true for self-organizing AND and XOR.

4.2 Circuits of Self-Organizing Gates

As an example of a circuit of self-organizing gates, consider a self-organizing 2-bit by 2-bit multiplier. Its schematics is presented in Fig. 7(a). The self-organizing multiplier involves eight self-organizing gates and eight voltage-controlled differential current generators. This circuit was designed based on the conventional 2-bit by 2-bit binary multiplier using the circuit transformation rules outlined in Sec. 2.

The circuit in Fig. 7(a) uses four constant voltage sources V3–V6 to encode the number to factorize (it is 6 in Fig. 7(a)). We emphasize that the signals P0–P3 serve as the input. The output signals are A0, A1, B0, and B1, which are the bits of two factors (A0 and B0 are the least significant bits). These factors are found through the deterministic dynamics of the self-organizing multiplier.

Examples of circuit dynamics are demonstrated in Figs. 7(b)–(f). In particular, Figs. 7(b) and (c) show that the result may be different in different runs. Specifically, these graphs indicate that number 2 can be presented as $2 \cdot 1$ or $1 \cdot 2$. This ability to identify different solutions is related to the random choice of initial states of memristive elements. Two distinct solutions are observed when two sets of initial states belong to different basins of attraction.

Figures 7(e) and (f) indicate that the factorization of some numbers can be more difficult than others (using some close initial conditions). In our simulations of the self-organizing multiplier, the most difficult was the factorization of 1. In this case, most frequently, we have observed the transition to a limit cycle behavior as the one in Fig. 7(f) and quite occasionally the correct solution to the problem ($1 = 1 \cdot 1$).

We have verified that the existence of the limit cycle (Fig. 7(f)) is not related to certain modifications to the parameters that we made. In particular, the limit cycle was observed in the circuit without C1 (in the VCDCG model) and with prior values of R_{on}, q, and i_{max} (from Tab. A2). In a longer simulation, it was observed that the limit cycle continues up to 100 s. To ensure that the limit cycle is not a numerical artifact, we performed some additional simulations. The use of other numerical integration methods in LTspice (trapezoid and modified trap in addition to Gear)[2], variation of tolerances, noise addition, and the use of PSPICE result in the same limit cycle behavior.

The existence of the limit cycle in the numerical dynamics seems to contradict the statement "if the Boolean problem the DMMs are designed to solve has a solution, the system will always find it, irrespective of the initial conditions" in [4] (see also [1]) for the continuous dynamics. Currently, the exact reason for this is not known, and its determination is beyond the scope of this work.

5. Conclusion

Having identified and corrected some inconsistencies in the prior literature [1, 2, 32] (see Items 1–4 in Sec. 3.2), we have formulated SPICE models of the Design I self-organizing logic gates. The operation of individual self-organizing gates and small circuits thereof has been demonstrated.

[2]It is known that the basin of attraction is influenced by the discretization [34].

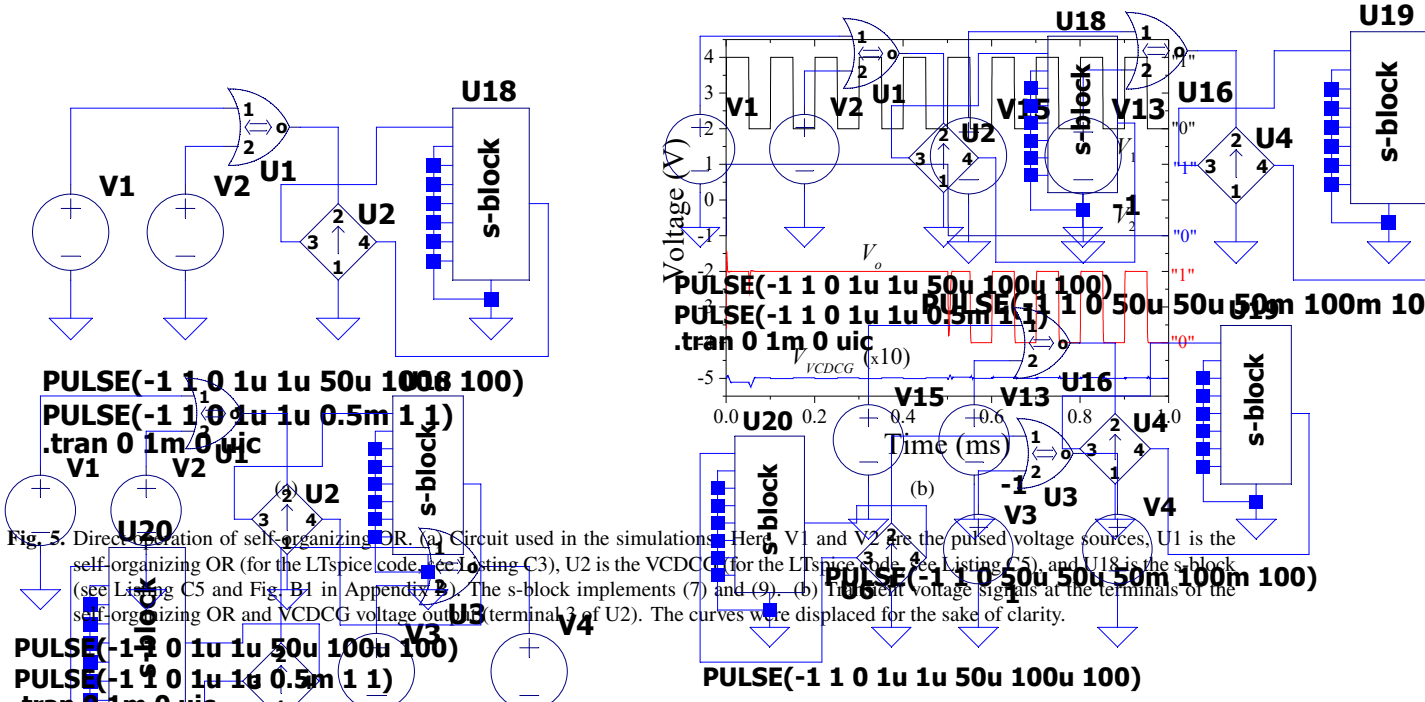


Fig. 5. Direct operation of self-organizing OR. (a) Circuit used in the simulations. Here, V1 and V2 are the pulsed voltage sources, U1 is the self-organizing OR (for the LTspice code, see Listing C3), U2 is the VCDCG (for the LTspice code, see Listing C5), and U18 is the s-block (see Listing C5 and Fig. B1 in Appendix B). The s-block implements (7) and (9). (b) Transient voltage signals at the terminals of the self-organizing OR and VCDCG voltage output (terminal 3 of U2). The curves were displaced for the sake of clarity.

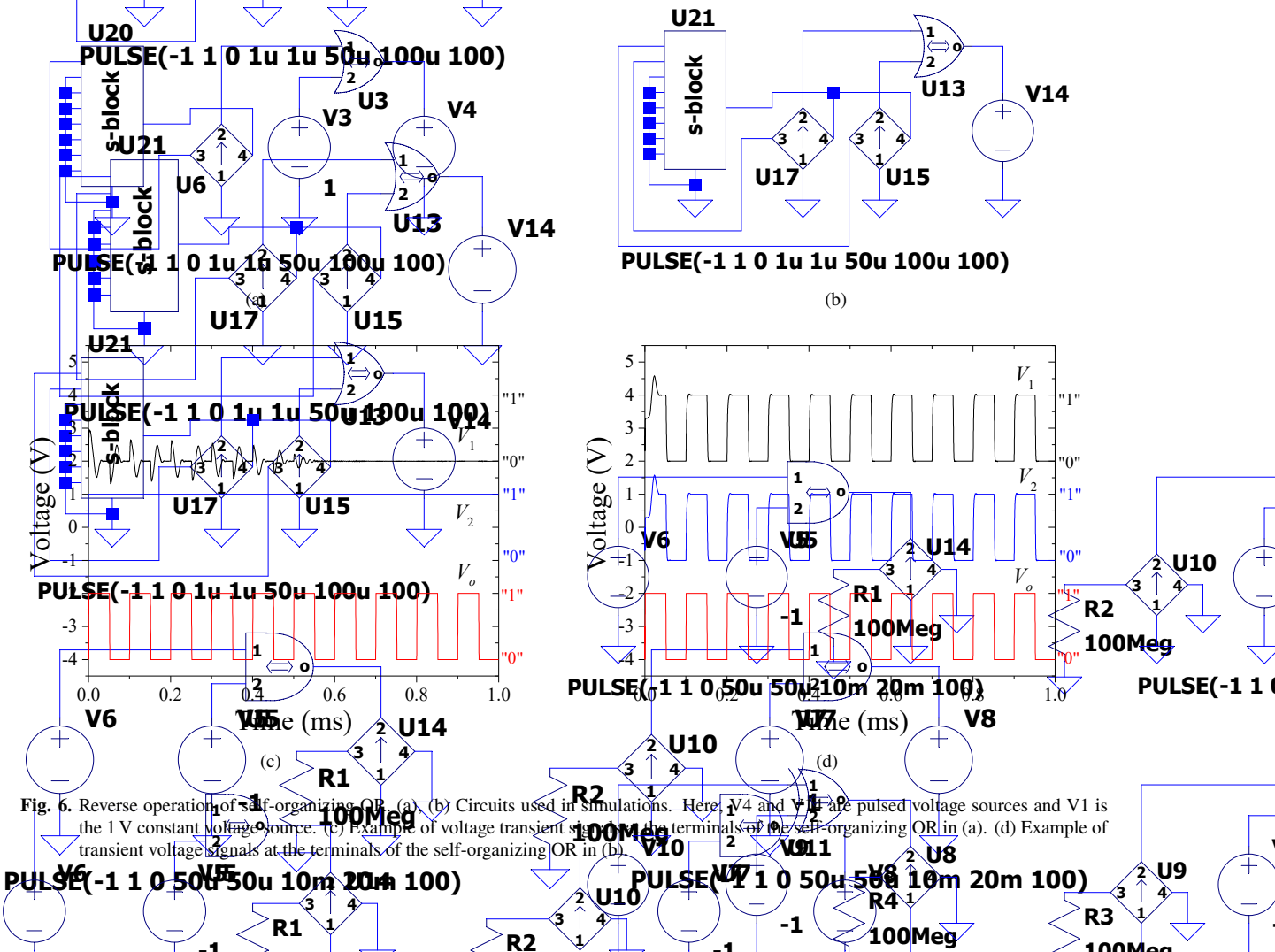
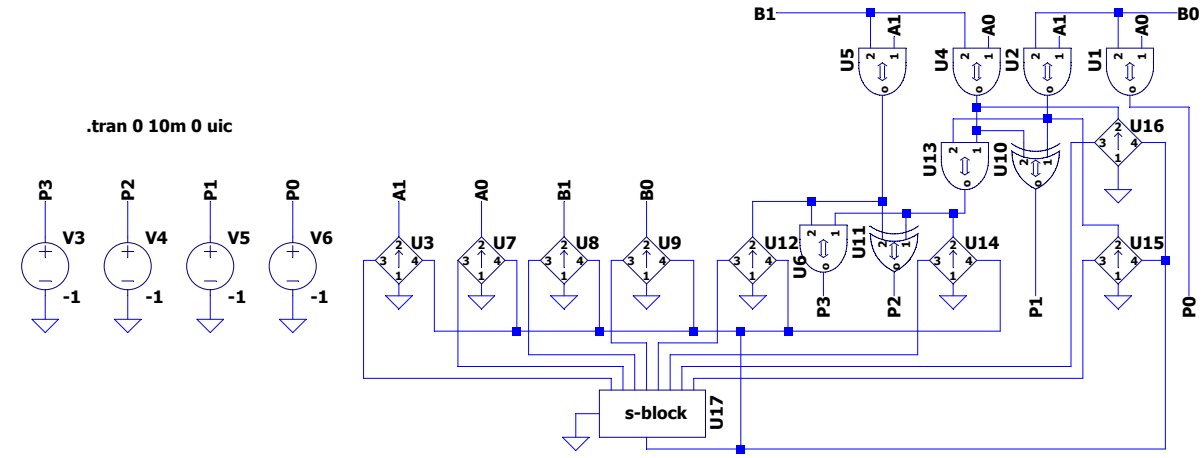
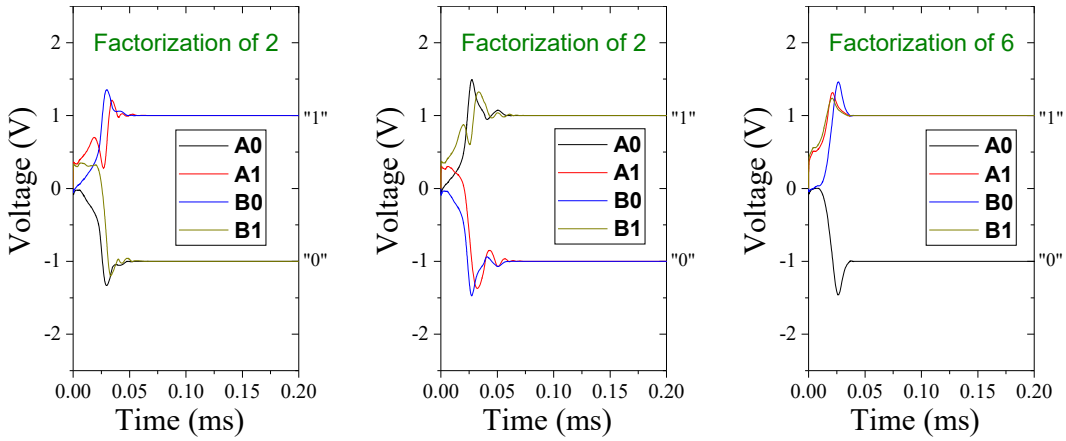


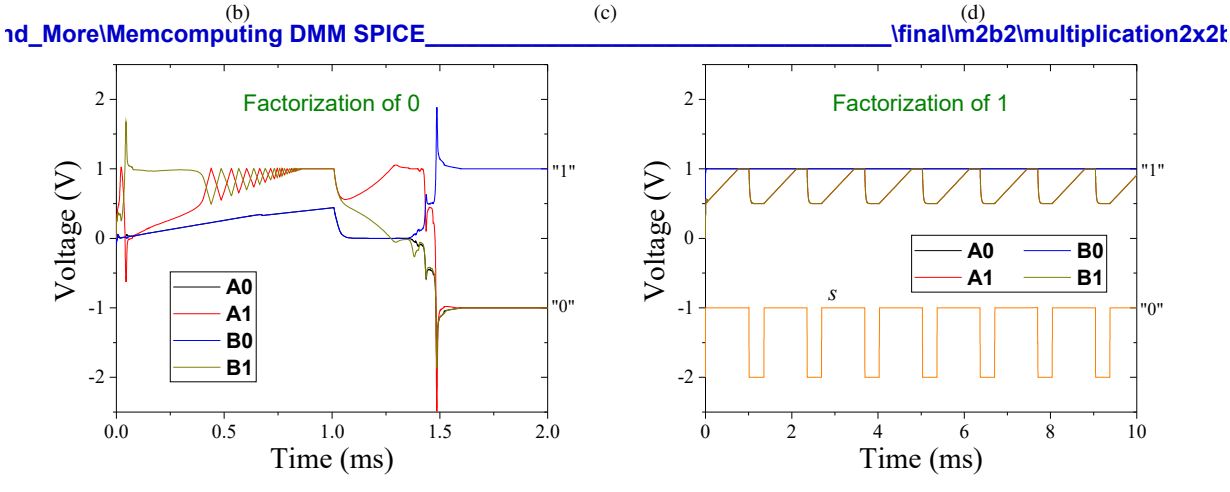
Fig. 6. Reverse operation of self-organizing OR. (a), (b) Circuits used in simulations. Here, V4 and V14 are pulsed voltage sources and V1 is the 1 V constant voltage source. (c) Example of voltage transient signals at the terminals of the self-organizing OR in (a). (d) Example of transient voltage signals at the terminals of the self-organizing OR in (b).



(a)



(b) (c) (d)



(e) (f)

Fig. 7. Solving integer factorization problem with a self-organizing 2-bit by 2-bit multiplier. (a) Circuit used in the simulations. Here, the number to factorize is represented by (P3,P2,P1,P0), and the factors – by (A1,A0) and (B1,B0). (b)–(f) Examples of the transient dynamics of the self-organizing 2-bit by 2-bit multiplier. Curve *s* in (f) was shifted down by 2 V for clarity.

We emphasize that in future studies of these gates, special attention should be paid among others to:

- Polarity of memristive devices in DCMs.
- Use of resistors in the schematics of the universal gate.
- Sign of the k_i term in the function $f_s\left(\mathbf{i}_{\mathrm{DCG}}, s_j\right)$.
- Values for parameters k_i and k_s.

In summary, self-organizing logic gates are an interesting generalization of the traditional Boolean logic gates. The SPICE models reported in this paper offer an easy and pretty reliable way to explore self-organization in memcomputing circuits. Experimental demonstration of self-organizing gates is an interesting project for the future.

Acknowledgments

The author acknowledges the support of the National Science Foundation grant number ECCS-2229880. He is thankful to M. Di Ventra, D. C. Nguyen, and Y.-H. Zhang for numerous discussions on various aspects of memcomputing.

References

[1] TRAVERSA, F. L., DI VENTRA, M. Polynomial-time solution of prime factorization and NP-complete problems with digital memcomputing machines. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 2017, vol. 27, no. 2, p. 1–22. DOI: 10.1063/1.4975761

[2] DI VENTRA, M. *MemComputing: Fundamentals and Applications*. 1st ed. Oxford (UK): Oxford University Press, 2022. ISBN: 9780192659897

[3] DI VENTRA, M., TRAVERSA, F. L. Absence of chaos in digital memcomputing machines with solutions. *Physics Letters A*, 2017, vol. 381, no. 38, p. 3255–3257. DOI: 10.1016/j.physleta.2017.08.040

[4] DI VENTRA, M., TRAVERSA, F. L. Absence of periodic orbits in digital memcomputing machines with solutions. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 2017, vol. 27, no. 10, p. 1–1. DOI: 10.1063/1.5004431

[5] DI VENTRA, M., TRAVERSA, F. L., OVCHINNIKOV, I. V. Topological field theory and computing with instantons. *Annalen der Physik*, 2017, vol. 529, no. 12, p. 1–6. DOI: 10.1002/andp.201700123

[6] DI VENTRA, M., OVCHINNIKOV, I. V. Digital memcomputing: From logic to dynamics to topology. *Annals of Physics*, 2019, vol. 409, p. 1–12. DOI: 10.1016/j.aop.2019.167935

[7] KOWALSKY, M., ALBASH, T., HEN, I., et al. 3-regular three-XORSAT planted solutions benchmark of classical and quantum heuristic optimizers. *Quantum Science and Technology*, 2022, vol. 7, no. 2, p. 1–18. DOI: 10.1088/2058-9565/ac4d1b

[8] GOTO, H., TATSUMURA, K., DIXON, A. R. Combinatorial optimization by simulating adiabatic bifurcations in nonlinear Hamiltonian systems. *Science Advances*, 2019, vol. 5, no. 4, p. 1–9. DOI: 10.1126/sciadv.aav2372

[9] MATSUBARA, S., TAKATSU, M., MIYAZAWA, T., et al. Digital annealer for high-speed solving of combinatorial optimization problems and its applications. In *25th Asia and South Pacific Design Automation Conference (ASP-DAC)*. Beijing (China), 2020, p. 667–672. DOI: 10.1109/ASP-DAC47756.2020.9045100

[10] BEARDEN, S. R. B., MANUKIAN, H., TRAVERSA, F. L., et al. Instantons in self-organizing logic gates. *Physical Review Applied*, 2018, vol. 9, no. 3, p. 1–8. DOI: 10.1103/PhysRevApplied.9.034029

[11] CHUA, L. O., KANG, S. M. Memristive devices and systems. *Proceedings of IEEE*, 1976, vol. 64, no. 2, p. 209–223. DOI: 10.1109/PROC.1976.10092

[12] DI VENTRA, M., PERSHIN, Y. V., CHUA, L. O. Circuit elements with memory: Memristors, memcapacitors, and meminductors. *Proceedings of the IEEE*, 2009, vol. 97, no. 10, p. 1717–1724. DOI: 10.1109/JPROC.2009.2021077

[13] PERSHIN, Y. V., DI VENTRA, M. Memory effects in complex materials and nanoscale systems. *Advances in Physics*, 2011, vol. 60, no. 2, p. 145–227. DOI: 10.1080/00018732.2010.544961

[14] OCHS, K., MICHAELIS, D., SOLAN, E. Towards wave digital memcomputing with physical memristor models. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 2020, vol. 67, no. 4, p. 1092–1102. DOI: 10.1109/TCSI.2019.2953653

[15] BEARDEN, S. R. B., PEI, Y. R., DI VENTRA, M. Efficient solution of Boolean satisfiability problems with digital memcomputing. *Scientific Reports*, 2020, vol. 10, no. 1, p. 1–8. DOI: 10.1038/s41598-020-76666-2

[16] ERCSEY-RAVASZ, M., TOROCZKAI, Z. Optimization hardness as transient chaos in an analog approach to constraints satisfaction. *Nature Physics*, 2011, vol. 7, no. 12, p. 966–970. DOI: 10.1038/NPHYS2105

[17] MOLNÁR, F., KHAREL, S. R., HU, X. S., et al. Accelerating a continuous-time analog SAT solver using GPUs, *Computer Physics Communications*, 2020, vol. 256, p. 1–14. DOI: 10.1016/j.cpc.2020.107469

[18] NGUYEN, D. C., ZHANG, Y.-H., DI VENTRA, M. et al. Hardware implementation of digital memcomputing on small-size FPGAs. In *Proceedings of IEEE 66th International Midwest Symposium on Circuits and Systems (MWSCAS)*. Phoenix (Arizona, USA), 2023, arXiv:2305.01061. [Forthcoming]

[19] DI VENTRA, M., PERSHIN, Y. V. *Memristors and Memelements: Mathematics, Physics, and Fiction*. 1st ed. Cham (Switzerland): Springer, 2023. ISBN: 9783031256240

[20] BIOLEK, Z., BIOLEK, D., BIOLKOVA, V. SPICE modeling of memristive, memcapacitative and meminductive systems. In *Proceedings of European Conference on Circuit Theory and Design (ECCTD)*. Antalya (Turkey), 2009, p. 249–252. DOI: 10.1109/ECCTD.2009.5274934

[21] PERSHIN, Y. V., DI VENTRA, M. SPICE model of memristive devices with threshold. *Radioengineering*, 2013, vol. 22, no. 2, p. 485–489. ISSN: 1210-2512

[22] BIOLEK, D., DI VENTRA, M., PERSHIN, Y. V. Reliable SPICE simulations of memristors, memcapacitors and meminductors. *Radioengineering*, 2013, vol. 22, no. 4, p. 945–968. ISSN: 1210-2512

[23] XU, K. D., ZHANG, Y. H., WANG, L., et al. Two memristor SPICE models and their applications in microwave devices. *IEEE Transactions on Nanotechnology*, 2014, vol. 13, no. 3, p. 607–616. DOI: 10.1109/TNANO.2014.2314126

[24] VOURKAS, I., BATSOS, A., SIRAKOULIS, G. C. SPICE modeling of nonlinear memristive behavior. *International Journal of Circuit Theory and Applications*, 2015, vol. 43, no. 5, p. 553–565. DOI: 10.1002/cta.1957

[25] LI, Q., SERB, A., PRODROMAKIS, T., et al. A memristor SPICE model accounting for synaptic activity dependence. *PloS One*, 2015, vol. 10, no. 3, p. 1–12. DOI: 10.1371/journal.pone.0120506

[26] BIOLEK, D., KOLKA, Z., BIOLKOVA, V., et al. Memristor models for SPICE simulation of extremely large memristive networks. In *Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS)*. Montreal (QC, Canada), 2016, p. 389–392. DOI: 10.1109/ISCAS.2016.7527252

[27] GARCIA-REDONDO, F., GOWERS, R. P., CRESPO-YEPES, A., et al. SPICE compact modeling of bipolar/unipolar memristor switching governed by electrical thresholds. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 2016, vol. 63, no. 8, p. 1255–1264. DOI: 10.1109/TCSI.2016.2564703

[28] SCHROEDTER, R., DEMIRKOL, A. S., ASCOLI, A., et al. SPICE compact model for an analog switching niobium oxide memristor. In: *Proceedings of 11th International Conference on Modern Circuits and Systems Technologies (MOCAST)*. Bremen (Germany), 2022, p. 1–4. DOI: 10.1109/MOCAST54814.2022.9837726

[29] DOWLING, V. J., SLIPKO, V. A., PERSHIN, Y. V. Analytic and SPICE modeling of stochastic ReRAM circuits. In *SPIE Proceedings of International Conference on Micro- and Nano-Electronics*. Zvenigorod (Russia), 2021, p. 1–9. DOI: 10.1117/12.2624571

[30] VLADIMIRESCU, A. *The SPICE Book*. 1st ed. New York (USA): Wiley, 1994. ISBN: 9780471609261

[31] KUNDERT, K. *The Designer's Guide to SPICE and SPECTRE*. New York (USA): Kluwer Academic Publishers, 1995. ISBN: 9780792395713

[32] DI VENTRA, M., TRAVERSA, F. L. *Self-Organizing Logic Gate and Circuits and Complex Problem Solving with Self-Organizing Circuits*. US Patent 9911080, 2018. Available at: https://patents.google.com/patent/US9911080

[33] STRUKOV, D. B., SNIDER, G. S., STEWART, D. R., et al. The missing memristor found. *Nature*, 2008, vol. 453, no. 7191, p. 80–83. DOI: 10.1038/nature06932

[34] ZHANG, Y.-H., DI VENTRA, M. Directed percolation and numerical stability of simulations of digital memcomputing machines. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 2021, vol. 31, no. 6, p. 1–12. DOI: 10.1063/5.0045375

About the Authors . . .

Yuriy V. PERSHIN was born in 1973. He received his M.Sc. degree from Kharkov State University, Ukraine, in 1996 and his Ph.D. degree from the University of Konstanz, Germany, in 2002. After several postdoctoral positions, he joined the faculty at the University of South Carolina in 2008. In 2014, he was promoted to the rank of Associate Professor. His research interests include unconventional computing, emerging memory devices, and quantum materials.

Appendix A: Parameters

	Terminal 1				Terminal 2				Out Terminal			
	a_1	a_2	a_0	dc	a_1	a_2	a_0	dc	a_1	a_2	a_0	dc
SO AND												
L_{M_1}	0	–1	1	v_c	–1	0	1	v_c	1	0	0	0
L_{M_2}	1	0	0	0	0	1	0	0	0	1	0	0
L_{M_3}	0	0	1	0	0	0	1	0	0	0	1	0
L_{M_4}	1	0	0	0	0	1	0	0	2	2	–1	$-2v_c$
L_R	4	1	–3	$-v_c$	1	4	–3	$-v_c$	–4	–4	7	$2v_c$
SO OR												
L_{M_1}	0	0	1	0	0	0	1	0	0	0	1	0
L_{M_2}	1	0	0	0	0	1	0	0	2	2	–1	$2v_c$
L_{M_3}	0	–1	1	$-v_c$	–1	0	1	$-v_c$	1	0	0	0
L_{M_4}	1	0	0	0	0	1	0	0	0	1	0	0
L_R	4	1	–3	v_c	1	4	–3	v_c	–4	–4	7	$-2v_c$
SO XOR												
L_{M_1}	0	–1	–1	v_c	–1	0	–1	v_c	–1	–1	0	v_c
L_{M_2}	0	1	1	v_c	1	0	1	v_c	1	1	0	v_c
L_{M_3}	0	–1	1	$-v_c$	–1	0	1	$-v_c$	–1	1	0	$-v_c$
L_{M_4}	0	1	–1	$-v_c$	1	0	–1	$-v_c$	1	–1	0	$-v_c$
L_R	6	0	–1	0	0	6	–1	0	–1	–1	7	0

Tab. A1. Parameters of voltage-controlled voltage generators for AND, OR, and XOR gates.

Parameter	Value	Parameter	Value	Parameter	Value
R_{on}	10^{-2}	R_{off}	1	v_c	1
α	60	C	10^{-9}	k	∞
V_t	0	γ	60	q	10
m_0	–400	m_1	400	i_{min}	10^{-8}
i_{max}	20	k_i	10^{-7}	k_s	10^{-7}
δ_s	0	δ_i	0		

Tab. A2. Parameters of numerical simulations used in [1].

Appendix B: s-block

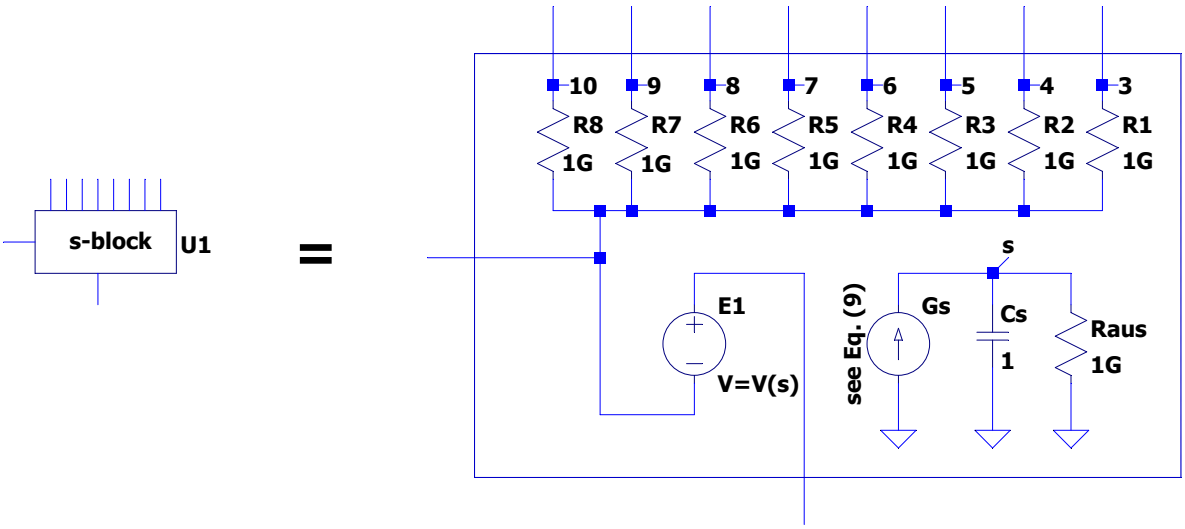


Fig. B1. Schematics of s-block. The s-block implements (7) and (9) according to the LTspice model in Listing C5.

Appendix C: SPICE Models

Listing C1. LTspice code for the self-organizing AND.

```
****Self-organizing AND ************************
****Code for LTspice; tested with LTspice XVII *********
***********************************************
.subckt SAND 1 2 3
.param zI=1E5 ; scaling factor
.param res={1*zI} vc=1
*DCM1
EM11 11 0 value={-V(2)+V(3)+vc}
Xmem11 1 11 memR
EM14 14 0 value={V(3)}
Xmem14 14 1 memR
EM13 13 0 value={4*V(1)+1*V(2)-3*V(3)-vc}
R11 13 1 {res}
*DCM2
EM21 21 0 value={-V(1)+V(3)+vc}
Xmem21 2 21 memR
EM24 24 0 value={V(3)}
Xmem24 24 2 memR
EM23 23 0 value={V(1)+4*V(2)-3*V(3)-vc}
R21 23 2 {res}
*DCM3
EM31 31 0 value={V(1)}
Xmem31 3 31 memR
EM32 32 0 value={V(2)}
Xmem32 3 32 memR
EM35 35 0 value={2*V(1)+2*V(2)-1*V(3)-2*vc}
Xmem35 35 3 memR
EM33 33 0 value={-4*V(1)-4*V(2)+7*V(3)+2*vc}
R31 33 3 {res}
*resistors
R1o 1 3 {res}
R2o 2 3 {res}
.ends SAND
```

Listing C2. LTspice code for the self-organizing XOR.

```
****Self-organizing XOR ***************************
****Code for LTspice; tested with LTspice XVII **********
************************************************
.subckt SXOR 1 2 3
.param zI=1E5 ; scaling factor
.param res={1*zI} vc=1
*DCM1
EM11 11 0 value={-V(2)-V(3)+vc}
Xmem11 1 11 memR
EM12 12 0 value={V(2)+V(3)+vc}
Xmem12 1 12 memR
EM14 14 0 value={-V(2)+V(3)-vc}
Xmem14 14 1 memR
EM15 15 0 value={V(2)-V(3)-vc}
Xmem15 15 1 memR
EM13 13 0 value={6*V(1)-V(3)}
R11 13 1 {res}
*DCM2
EM21 21 0 value={-V(1)-V(3)+vc}
Xmem21 2 21 memR
EM22 22 0 value={V(1)+V(3)+vc}
Xmem22 2 22 memR
EM24 24 0 value={-V(1)+V(3)-vc}
Xmem24 24 2 memR
EM25 25 0 value={V(1)-V(3)-vc}
Xmem25 25 2 memR
EM23 23 0 value={6*V(2)-V(3)}
R21 23 2 {res}
*DCM3
EM31 31 0 value={-V(1)-V(2)+vc}
Xmem31 3 31 memR
EM32 32 0 value={V(1)+V(2)+vc}
Xmem32 3 32 memR
EM34 34 0 value={-V(1)+V(2)-vc}
Xmem34 34 3 memR
EM35 35 0 value={V(1)-V(2)-vc}
Xmem35 35 3 memR
EM33 33 0 value={-V(1)-V(2)+7*V(3)}
R31 33 3 {res}
*resistors
R1o 1 3 {res}
R2o 2 3 {res}
.ends SXOR
```

Listing C3. LTspice code for the self-organizing OR.

```
****Self-organizing OR ****************************
****Code for LTspice; tested with LTspice XVII *********
***********************************************
.subckt SOR 1 2 3
.param zI=1E5 ; scaling factor
.param res={1*zI} vc=1
*DCM1
EM11 11 0 value={V(3)}
Xmem11 1 11 memR
EM14 14 0 value={-V(2)+V(3)-vc}
Xmem14 14 1 memR
EM13 13 0 value={4*V(1)+1*V(2)-3*V(3)+vc}
R11 13 1 {res}
*DCM2
EM21 21 0 value={V(3)}
Xmem21 2 21 memR
EM24 24 0 value={-V(1)+V(3)-vc}
Xmem24 24 2 memR
EM23 23 0 value={V(1)+4*V(2)-3*V(3)+vc}
R21 23 2 {res}
*DCM3
EM32 32 0 value={2*V(1)+2*V(2)-V(3)+2*vc}
Xmem32 3 32 memR
EM34 34 0 value={V(1)}
Xmem34 34 3 memR
EM35 35 0 value={V(2)}
Xmem35 35 3 memR
EM33 33 0 value={-4*V(1)-4*V(2)+7*V(3)-2*vc}
R31 33 3 {res}
*resistors
R1o 1 3 {res}
R2o 2 3 {res}
.ends SOR
```

Listing C4. LTspice code for the memristive element.

```
****Memristive element ***********************************************
****Code for LTspice; tested with LTspice XVII ****************************
*********************************************************************
.subckt memR plus minus
.param zI=1E5 zt=1E3 ; scaling factors
.param Ron={0.05*zI} Roff={1*zI} alpha={60*zI*zt} C={1E-9/zI/zt}
*model of memristive port
Gpm plus minus value={V(plus,minus)/(Ron+(Roff-Ron)*V(x))}
C1 plus minus {C} IC={0}
*integrator model
Gx 0 x value={-alpha*h(V(x),V(plus,minus))*V(plus,minus)/(Ron+(Roff-Ron)*V(x))}
Cx x 0 1 IC={mc(0.2,0.1)} ; randomized initial condition
Raux x 0 1G
*functions
.func h(x,vm)={u(x)*u(vm)+u(1-x)*u(-vm)}
.ends memR
```

Listing C5. LTspice code for VCDCG and s-block.

```
****Voltage-Controlled Differential Current Generator ************************
****Code for LTspice; tested with LTspice XVII ****************************
********************************************************************
.subckt VCDCG 1 2 3 4
*1: negative terminal (ground); 2: positive terminal;
*3: current output signal (voltage); 4: input from the s-circuit
.param zI=1E5 zt=1E3 ; scaling factors
.param m0={-400/zI*zt} m1={400/zI*zt} q={10/zI*zt} gamma={60*zt} cap={1E-3/zI/zt}
*model of VCDCG ports
G1 2 1 value={V(x)}
E1 3 1 value={V(x)}
R1 4 1 1G
C1 2 1 {cap} IC={0}
*integrator model
Gx 0 x value = {u(V(4,1)-0.5)*fdcg(V(2,1))-gamma*u(0.5-V(4,1))*V(x)}
Cx x 0 1 IC={0}
Raux x 0 1G
*functions
.param m0b={m0*pi/(2*q)} m1b={m1*pi/(2*q)}
.func fdcg(x)={q*(atan(m1b*(x+1))+atan(m0b*x)+atan(m1b*(x-1)))*2/pi}
.ends VCDCG


****8-input s-block ***********************************************************
****Code for LTspice; tested with LTspice XVII ********************************
********************************************************************************
.subckt SCOMB8 1 2 3 4 5 6 7 8 9 10
*1: negative terminal (ground); 2: positive terminal (output); 3-10: inputs
.param zI=1E5 zt=1E3 ; scaling factors
.param imin={1E-8/zI} imax={10/zI} ks={2E3*zt} ki={2E3*zt}
*model
E1 2 1 value={V(s)}
R1 1 3 1G
R2 1 4 1G
R3 1 5 1G
R4 1 6 1G
R5 1 7 1G
R6 1 8 1G
R7 1 9 1G
R8 1 10 1G
*integrator model
Gs 0 s value ={-ks*V(s)*(V(s)-1)*(2*V(s)-1)-
+ki*(1-u(imin-abs(V(3,1)))*u(imin-abs(V(4,1)))*u(imin-abs(V(5,1)))*u(imin-abs(V(6,1)))*
+u(imin-abs(V(7,1)))*u(imin-abs(V(8,1)))*u(imin-abs(V(9,1)))*u(imin-abs(V(10,1)))-u(imax-abs(V(3,1)))*
+u(imax-abs(V(4,1)))*u(imax-abs(V(5,1)))*u(imax-abs(V(6,1)))*u(imax-abs(V(7,1)))*u(imax-abs(V(8,1)))*
+u(imax-abs(V(9,1)))*u(imax-abs(V(10,1))))}
Cs s 0 1 IC={0.75}
Raus s 0 1G
.ends SCOMB8
```