BioNanoSci. (2016) 6:214–234
DOI 10.1007/s12668-016-0213-1

Memristor-Based Volistor Gates Compute Logic with Low Power Consumption

Muayad Aljafar[1] · **Paul Long**[1] · **Marek Perkowski**[1]

Published online: 15 August 2016

Abstract We introduce a novel volistor logic gate which uses voltage as input and resistance as output. Volistors rely on the diode-like behavior of rectifying memristors. We show how to realize the first logic level, counted from the input, of any Boolean function with volistor gates in a memristive crossbar network. Unlike stateful logic, there is no need to store the inputs as resistances, and computation is performed directly. The fan-in and fan-out of volistor gates are large and different from traditional memristor circuits. Compared to solely memristive stateful logic, a combination of volistors and stateful inhibition gates can significantly reduce the number of operations required to calculate arbitrary multi-output Boolean functions. The power consumption of volistor logic is computed and compared with the power consumption of stateful logic using the simulation results obtained by LTspice—when implemented in a 1 × 8 or an 8 × 1 crosspoint array, volistors consume significantly less power.

Keywords Memristive Crosspoint array · Logic computation · Memristive crossbar array · Rectifying memristor · Stateful inhibition · Volistor logic

✉ Muayad Aljafar
muayad@pdx.edu

Paul Long
paul@thelongs.ws

Marek Perkowski
mperkows@ee.pdx.edu

[1] Department of Electrical and Computer Engineering, Portland State University, Portland, OR 97201, USA

1 Introduction

Stateful logic computation with memristors is an area of active research. Borghetti et al. [1] proposed realizing stateful logic via material implication (IMP). In classical stateful memristive circuits, logic signals utilize resistances on inputs and outputs, meaning, the previous resistance state of the memristor affects the operations. In contrast, volistors do not use the previous resistance state in calculations. Other stateful logic gates have been proposed as well, e.g., inhibition (INH) [2] or AND [3]. INH gate implements Boolean function $a\overline{b}$ where a and b are positive and negative inputs, respectively. In stateful logic, the logic values are encoded by the resistance states of memristors. Stateful logic is usually performed in a generic structure called a crossbar array. Leakage pathways due to the half-select of memristors in a crossbar row or column can be suppressed by using rectifying memristors [4]. One key disadvantage of stateful logic is that a long sequence of operations is required to implement an arbitrary Boolean function.

Memristor ratioed logic (MRL) [5] is another approach to logic computation and is based on the resistive ratio of non-rectifying memristors. In MRL, logic values are voltage-based pulses and circuit structures are dependent entirely on the Boolean function being implemented. The output voltage depends only on input voltages regardless of the resistance of the memristors; however, the resistances affect the propagation delay of the gate. MRL gates consume both dynamic and static power. Further, as it lacks the inversion function, MRL is not logically complete without CMOS inverters.

In this paper, we introduce a new concept in memristor logic. We call it volistor gate (voltage-resistor gate) which has voltage-based inputs and resistance-based outputs.

Therefore, volistor gates can only be used in the first level of logic implementation. This level can be composed of various types of gates and be complex. Logic synthesis methods with volistors are therefore different from classical logic synthesis. Volistors are implemented in generic crossbar arrays of rectifying memristors [6]. Unlike stateful logic, voltage pulses are the actual inputs to the volistor gates. It is always assumed that complemented input variables are available as voltage signals at no additional cost. Sometimes input signals are required in both positive and negative forms, e.g., x and $\bar{x}$. In addition, there is no propagation penalty as it exists in MRL. In a 1×8 crosspoint array, the propagation delay in volistors is shorter than in stateful INH. In a 1×8 crosspoint array, multi-input multi-output volistor gates consume less power than the corresponding stateful logic. The outputs of volistors are stored as the resistances of the target memristors. To provide the correct functionality, target memristors need to be initially closed, i.e., target memristors must be set to the low resistance state. With different coding schemes, either a volistor OR and NAND logic set or a volistor NOR and AND logic set can be realized in the same crossbar array. For instance, if the closed state of a memristor encodes logic "0" and the open state of a memristor encodes logic "1," the set of OR and NAND operations is implemented. The reverse encoding scheme implements the NOR and AND logic set. With volistor logic, crossbar drivers need to supply only three voltage levels: V^+, V^-, or 0 V. In addition, the control circuitry must be capable of setting arbitrary nanowires in the crossbar to either high impedance (HZ) or grounding these nanowires through load resistors R_G. Obviously, the volistor network layer cannot implement every Boolean function such as Sum of Products. However, a combination of a volistor NOR and AND with stateful INH, or their dual logic of volistor OR and NAND with stateful IMP, can both be used to realize arbitrary multi-output Boolean functions. As will be presented, this hybrid realization is faster than an equivalent circuit realized with only stateful gates. The speed comparison of hybrid and stateful logic circuits will be presented in Section 4.

In Section 2, the hysteresis behavior of rectifying memristors is reviewed. In Section 3, volistor gates are described. In Section 4, the synthesis of arbitrary Boolean functions is discussed. In Section 5, the power consumption of volistor gates is computed and compared with stateful logic. In section 6, volistors in memory application are discussed. Section 7 is a summary of the work.

2 Rectifying Memristors

We used a rectifying memristor [4, 6] as a linear bistable device [7]. The behavior of a rectifying memristor is defined in (1), following [7]. Equation (1) describes a simplified model of diode-like memristor M demonstrated practically in [4]. R_M denotes resistance of memristor M; s is the state variable of M normalized to a real number in the range 0 through 1, $s \in [0, 1]$, and therefore $R_{CLOSED} \le R_M \le R_{OPEN}$; v is the voltage applied across M.

$$R_M = \begin{cases} R_{OPEN}\left(\dfrac{R_{CLOSED}}{R_{OPEN}}\right)^s & v \ge 0 \\ R_{OPEN} & v < 0 \end{cases} \quad (1)$$

R_{OPEN} denotes high resistance state of memristor M; R_{CLOSED} is low resistance state of memristor M. It is assumed that $R_{OPEN} = 500M\Omega$ and $R_{CLOSED} = 500K\Omega$ which are consistent with empirical results reported in [4]. The dynamic behavior of the state variable s is such that s changes in time as described in the linear differential Eq. (2). In Equation (2), v_{CLOSE} is a positive threshold voltage; v_{OPEN} is a negative threshold voltage. For simplicity, it is assumed that the threshold voltages v_{CLOSE} and v_{OPEN} are symmetric ($v_{CLOSE} = -v_{OPEN}$) and $v_{CLOSE} = 1V$; α is a positive constant associated with the switching rate of the memristor. Here, α is assumed to be 125×10^7 $(V.s)^{-1}$, following [7].

$$\frac{ds}{dt} = \begin{cases} \alpha(v - v_{CLOSE}) & v > v_{CLOSE} \\ \alpha(v - v_{OPEN}) & v < v_{OPEN} \\ 0 & elsewhere \end{cases} \quad (2)$$

Figure 1a shows the $i - v$ characteristic of rectifying memristor M described in (1) and (2). In this work, the positive programming voltage is defined $v = 1.2V$ and denoted by V_{SET}; the application of V_{SET} will close switch M when it is open. Also, the negative programming voltage is defined $v = -1.2V$ and denoted by V_{CLEAR}; the application of V_{CLEAR} will open switch M when it is closed. Substituting all related values in Eq. (2) results in

$$\frac{ds}{dt} = \begin{cases} 0.2\alpha & v = 1.2 \\ -0.2\alpha & v = -1.2 \\ 0 & elsewhere \end{cases}$$

The straightforward analytical solution of this differential equation is

$$s(t+1) = \begin{cases} 0.2\alpha T^+ & v = 1.2 \\ -0.2\alpha T^- & v = -1.2 \\ s(t) & elsewhere \end{cases}$$

Since s is normalized to interval [0, 1], for the given values of α and v, the assumption of $s(t+1) = 1$ results in the switching delay of $T^+ = 4ns$. The desired solution for (2) is

$$s(t+1) = \begin{cases} 1 & v = 1.2 \\ 0 & v = -1.2 \\ s(t) & -1 \le v \le 1 \end{cases}$$

Note that $s(t)$ and $s(t+1)$ denote the current state and the next state of memristor M, respectively; for $-1 \le v \le 1$, the resistance state of memristor remains unchanged, i.e., $s(t+1) = s(t)$. Obviously, a larger V_{SET} results in a smaller switching delay T^+, as suggested by Eq. (2).

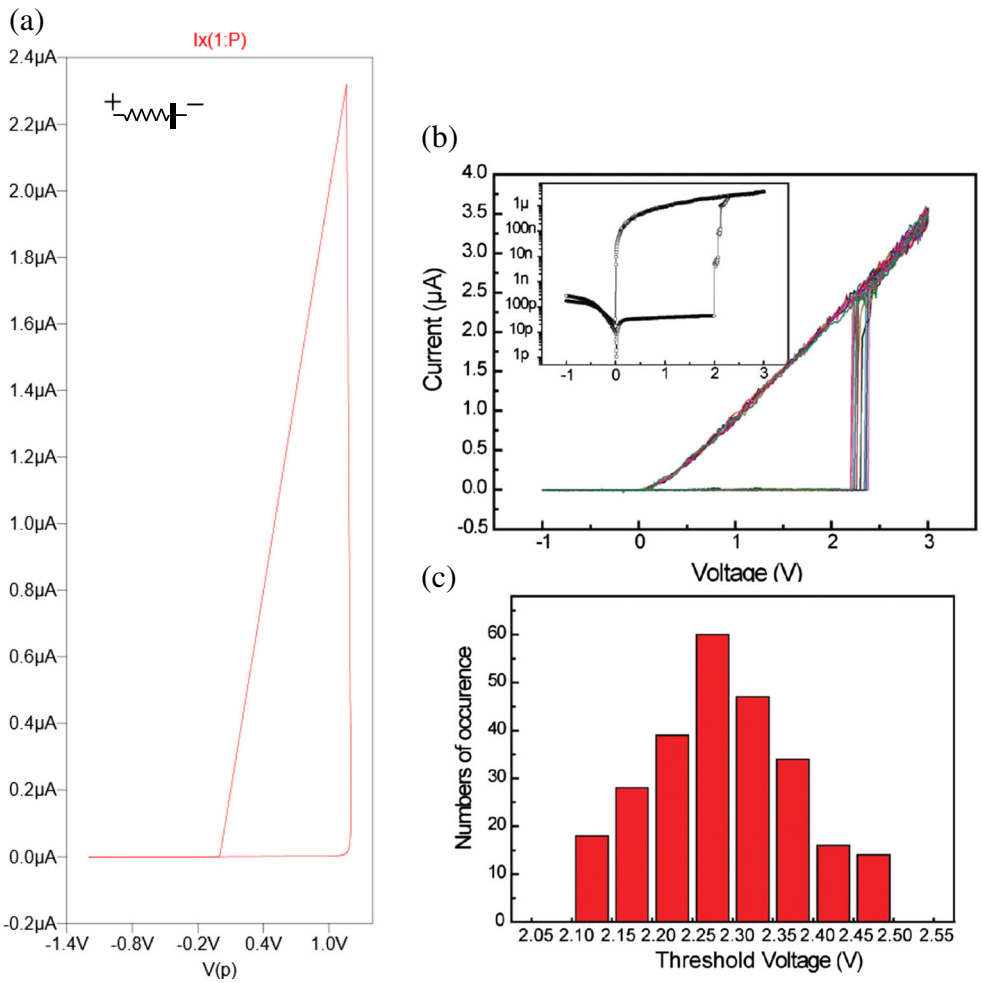


Fig. 1 **a** The i − v characteristic of rectifying memristor *M* described in (1) and (2). A sinusoidal input voltage with frequency 25 MHz and amplitude 1.2 V is applied to memristor M. The inset shows the symbolic diagram of a rectifying memristor. **b** The i − v characteristics of 10 different rectifying memristors in the crossbar array. The inset shows the i − v characteristics plotted in log scale demonstrating current suppression at negative bias in the on-state [4]. **c** Threshold voltage distribution of 256 cells in the fabricated crossbar array. The threshold voltage is defined as the voltage at which the measured current is above 10^{-6} A [4]

This simplified model does not accurately describe the behavior of rectifying memristor *M*. For instance, the threshold voltages v_{CLOSE} and v_{OPEN} are assumed to be constant, and/or the programming rate $\frac{ds}{dt}$ and the applied voltage v are assumed to be piecewise linearly related. The behavior of rectifying memristors described in (1) and (2) is compared with the actual behavior of memristors reported in [4, 6] for applied voltage v below.

1) $v < 0\mathrm{V}$

The actual behavior of rectifying memristors shows that $I^- < 10^{-13}$ A and $\frac{I^-}{I^+} < 10^{-6}$, where I^- is the reverse biased current and I^+ is the forward biased current, and the $\frac{R_{\mathrm{OPEN}}}{R_{\mathrm{CLOSED}}}$ ratio is from three to six orders of magnitude ($10^3 - 10^6$). In our model, the $\frac{R_{\mathrm{OPEN}}}{R_{\mathrm{CLOSED}}}$ ratio is 10^3 , which is consistent with the empirical results. However, the $\frac{I^-}{I^+}$ ratio is 10^{-3} which is much larger than the empirical results. Thus, using a precise memristor model would show even less power consumption in a crossbar array than our used model.

2) $0\mathrm{V} < v < v_{\mathrm{CLOSE}}$

Figure 1b shows the $i - v$ characteristics of ten rectifying memristors in a 16×16 crossbar array overlaid on top of each other. The memristors show approximately the same piecewise linear relationship between i and v in the interval $[0, v_{\mathrm{CLOSE}}]$. In other words, for applied voltage v, almost the same amount of current flows through all ten memristors. Therefore, Eqs. (1) and (2) with a fixed v can be applied uniformly to describe the behaviors exhibited by all memristors in the given interval. The linear behavior observed in Fig. 1a is the result of applying (1) and (2) which is uniform throughout all the memristors (Fig. 2).

3) $v > v_{\mathrm{CLOSE}}$

The threshold voltage distribution of the rectifying memristors in the crossbar array is shown in Fig. 1c [4]. The difference in the threshold voltages results in a difference in

the programming rates. Since the difference in the threshold voltages is not significant, i.e., $v_{CLOSE} \in [2.1V, 2.5V]$, the difference in the programming rates is not significant either. A small increase in the pulse width driving the crossbar array ensures a complete state transition in the memristors. In our model, the programming rate is assumed to be constant and in accordance with the empirical results reported in [8, 9]. Although the description of the behavior of the rectifying memristors modeled by (1) and (2) is imprecise, they still can be used in estimating the power consumption in CMOS-memristive circuits.

3 Volistor Logic

In this section, volistor logic is introduced. The key idea behind volistor logic is that inputs are voltages and outputs are resistances. This is a significant change from the way the voltage drivers are used in stateful logic; however, the target memristor is still used as a memory element where the output is stored. The basic volistor logic gates are inverter, *k*-input NOR and *k*-input AND, and their duals: *k*-input OR and *k*-input NAND. In the following subsections, we define the basic architecture required for logic computations and describe the basic logic gates.

3.1 Crosspoint Architecture

The basic circuit structures for volistor logic computations are the 1 × 2 crosspoint array and 2 × 1 crosspoint array depicted in Fig. 2. The crosspoint array is a horizontal or a vertical vector of memristors, i.e., a one-dimensional array. Figure 2a shows a generic structure for logic operations. The circuit is comprised of two rectifying memristors, labeled S and T, electrically connected through the common horizontal nanowire HL. Just as in stateful IMP, S denotes the source memristor and T the target memristor. The vertical nanowire VL1 connected to source memristor S conveys input signal V_{in}. Let $V^+ = 0.6V$ and $V^- = -0.6V$. The logical coding scheme for V_{in} is defined as follows: V^+ encodes logic "1," denoted $vin = 1$; 0 V encodes logic "0," denoted $vin = 0$. The vertical nanowire VL2 connected to target memristor T carries a bias voltage, $V_{bias} = V^-$. Memristor T acts as a switch whose resistivity state t represents the output of the crosspoint array. When T is open, its high-resistivity state encodes logic "0," i.e., $t = 0$. When T is closed, its low-resistivity state encodes logic "1," $t = 1$. This interpretation of the resistivity state of T is used for performing NOR/AND logic set; another interpretation will be discussed in Section 3.5. Prior to logic computation, both memristors S and T must be closed. To unconditionally close a memristor, it must be forward biased by V_{SET}. Unlike the target memristor, the source memristor S acts as a diode. The 1 × 2 crosspoint array must satisfy (3).

$$\begin{cases} V_{OPEN} < V_{bias} < 0V \\ V_{bias} - V_{in} = V_{CLEAR} \end{cases} \tag{3}$$

Each nanowire must be either driven by one of the voltages V_{in}, V_{bias}, and 0 V or terminated with high impedance HZ or grounded by load resistor R_G as shown in Fig. 3. All these connections are realized with CMOS switches shown symbolically in Fig. 3. R_G is defined as geometric mean of R_{OPEN} and R_{CLOSED}, $\sqrt{R_{OPEN} \cdot R_{CLOSED}} = 15M\Omega$. The crosspoint array operates by the simultaneous application of V_{in} and V_{bias} to S and T, respectively. Since $V_{in} > V_{bias}$, Ohm's Law requires a flow of current through the array. However, given the structure of the array, the application of these voltages forward biases S and reverse biases T thus suppressing the flow of current. This means that $V_{HL} \approx V_{in}$ where V_{HL} is the voltage on HL. If (3) is satisfied, the voltage across T will toggle that memristor, i.e., the new state of target memristor becomes $t = 0$.

$$\begin{cases} 0V < V_{bias} < V_{CLOSED} \\ V_{in} - V_{bias} = V_{CLEAR} \end{cases} \tag{4}$$

The crosspoint arrays shown in Fig. 2 can be scaled to 1 × n and n × 1 arrays, allowing for multi-input multi-output volistor logic functions. The 1 × n and n × 1 crosspoint arrays must satisfy (3) and (4), respectively. In these arrays, logic computation is achieved by implementing the wired OR function, i.e., V_{HL} or V_{VL} denotes the logical OR of inputs vin_1, …, vin_k where V_{VL} is the voltage on vertical nanowire VL and $1 \le k < n$.

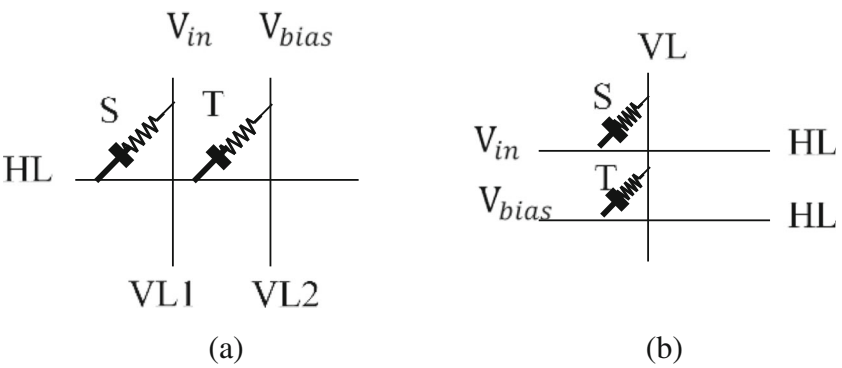


Fig. 2 **a** A 1 × 2 crosspoint array; **b** a 2 × 1 crosspoint array

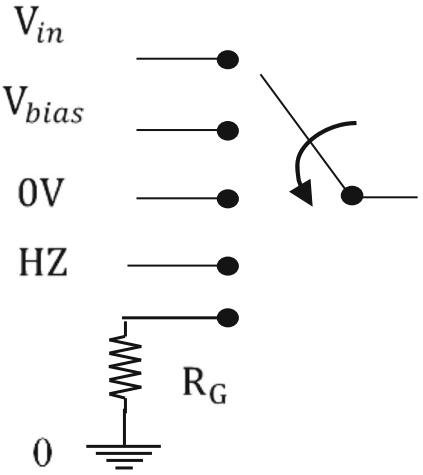


Fig. 3 The symbolic illustration of driver circuitry connected to each nanowire

3.2 Volistor NOT Gate in a Crosspoint Array

An inverter is the simplest gate to realize in volistor logic; the symbolic diagram is shown in Fig. 4a. Take, for example, the case where $(vin,\ t) = (1, 1)$ i.e., $vin = 1$ and $t = 1$. Both source and target memristors are initially closed. Single-output volistor NOT can be realized in either a 1×2 or a 2×1 crosspoint array. When realized in a 1×2 array, V_{bias} must be a negative voltage, V_{in} must be greater than or equal to 0 V, per (3), and horizontal nanowire HL must be connected to a high impedance, HZ. Connecting HL to HZ allows V_{in} to manifest on HL. Based on Ohm's Law, current should flow through the array, since $V_{in} - V_{bias} > 0$V. However, memristor T is reverse biased and thus suppresses the flow of current. In this case, the voltage drop across memristor S is 1.198 mV, i.e., the voltage on HL equals 598.801 mV. The voltage drop across T is 1.198 V which is sufficient to open memristor T. This is the desired behavior of the inverter function that it results in $(vin,\ t) = (1, 0)$. Given the parameters introduced in Section 2, the propagation delay of single-output volistor NOT is 4.044 ns as shown in Fig. 5b.

Single-output volistor NOT can be extended to an arbitrary fan-out which corresponds to a multi-output gate realized in arrays of types $1 \times n$ or $n \times 1$. The multi-output NOT gate stores $\overline{vin}$ in up to n-1 target memristors in any arbitrary location in the array. The role of each memristor is determined by its driver, i.e., source memristors are driven by V_{in} and target memristors by V_{bias}. Interestingly, this means that a memristor's function is independent of its position in the crosspoint array. Figure 5a shows a four-output volistor NOT gate implemented in a 1×8 array. Source memristor S is driven by V_{in} and target memristors $T_1 \cdots T_4$ are driven by V_{bias}. Memristors X_1, X_2 and X_3 are terminated to HZ and do not take part in the circuit's operation. The proper operation of a 63-output NOT gate with $vin = 1$ implemented in a 1×64 array and simulated in LTspice is depicted in Fig. 5c. V_{HL} is 528.881 mV and all 63 target memristors are switched off successfully in 6.223 ns. Figure 5d shows the desired behavior of volistor NOT implemented in a 1×2 array when $vin = 0$. $V_{HL} is\ -599.4\mu\ V$ and the target switch remains closed. Figure 5e shows the behavior of a sixty three-output volistor NOT implemented in a 1×64 array where $vin = 0$. This results in $V_{HL} = -35.559$ mV and all target switches remaining closed.

Table 1 summarizes these configurations and their effects on V_{HL}. A NOT with an arbitrary fan-out can be realized in one pulse. We discuss this gate for completeness; in practice, it is not required since the logical negation can be created by appropriately selecting the *vin* values. In this paper, all volistor gates can appear only at the input layer.

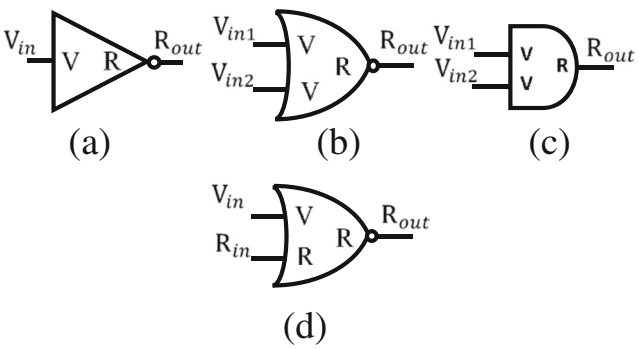


Fig. 4 Symbolic notation for volistor single-input and two-input logic gates. **a** Volistor inverter; **b** two-input volistor NOR gate; **c** two-input volistor AND gate; **d** mixed-input NOR gate. Inside the gates, symbols *V* and *R* denote whether a signal is a voltage-based or resistance-based, respectively

3.3 Volistor NOR Gate in a Crosspoint Array

The second basic gate in volistor logic is NOR. Figure 4b shows the symbolic diagram of a two-input volistor NOR gate. Since $1 \times n$ arrays have been previously discussed, in this subsection, $n \times 1$ arrays are considered. Take, for example, a two-input NOR gate where $(vin_1, vin_2, t) = (1, 0, 1)$. This requires the application of input data to S_1 and S_2 and V_{bias} to T, as shown in Fig. 6. Since the function is realized in a 3×1 crosspoint array, Eq. (4) requires $V_{bias} > 0$V and $V_{in} \leq 0$V. Based on Ohm's Law, since $V_{bias} - V_{in} > 0$, a current should flow through the crosspoint array.

However, S_2 and T are both reverse biased and thus they suppress the flow of current. The voltage drop across S_1 is 1.796 mV, i.e., $V_{VL} = -598.203$ mV. The voltage drop across T is −1.198 V which is sufficient to open T. Recall that all memristors are initially closed. The gate is realized by implementing wired-OR, i.e., the voltage on VL is the logical OR of vin_1 and vin_2. As before, V_{in} causes memristors S_1 and S_2 to behave as diodes and V_{bias} causes memristor T to behave as a

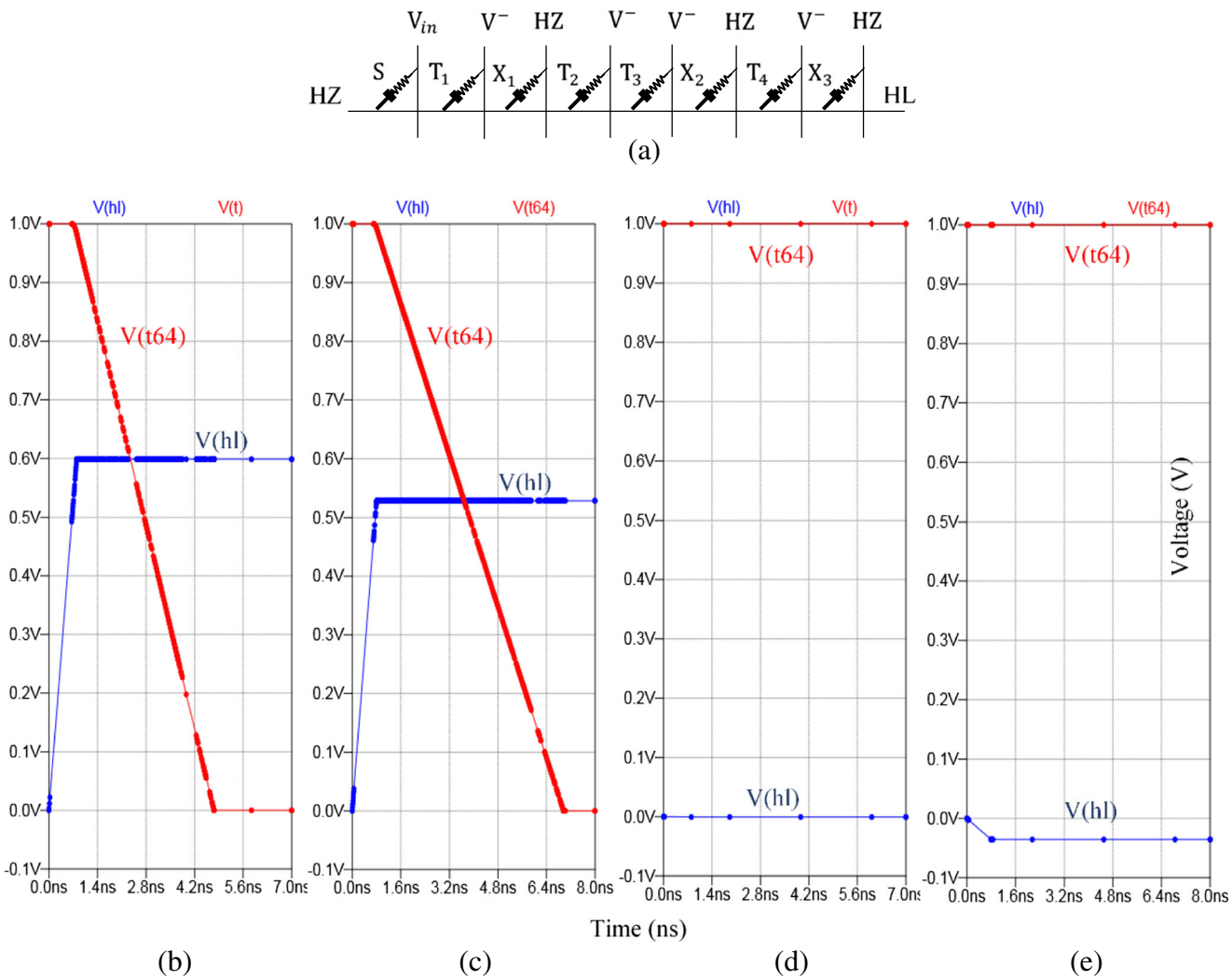


Fig. 5 Volistor NOT behavior. **a** A 1 × 8 crossbar array implementing a four-output NOT and showing arbitrary nature of the locations of *S* and *T*. The contribution of each memristor is determined by the voltage driver to which it is connected. The horizontal nanowire is connected to HZ. **b** The operation of a one output NOT in a 1 × 2 array. V(hl) stabilizes at ≈ 600 mV indicating *vin* = "1" manifesting on HL. In addition V(t), which is the SPICE model's representation of *t*, toggles to 0 V (*t* = "0"). **c** The operation of a 63 output NOT in a 1 × 64 array. V(hl) stabilizes and V(t) toggles as in **b. d** The operation of a one output NOT in a 1 × 2 array. V(hl) stabilizes at ≈−600 mV indicating *vin* = "0" manifesting on HL. As a result, V(t) remains 1 V (*t* = "1"). **e** V(hl) stabilizes as in **d**

switch. The output of the gate is the new state of the target switch T, either open or closed. Table 2 shows V_{VL} and t_d for various combinations of input values of a two-input, single-output volistor NOR gate implemented in a 3 × 1, and a sixty three-input, single-output volistor NOR gate implemented in a 64 × 1 crosspoint array. Note that with more logic "1" inputs, the value of V_{VL} approaches V_{in} and the propagation delay t_d gets shorter. Volistor logic allows a multi-input NOR gate to be realized in only one pulse.

Table 1 Implementation of multi-output volistor NOT gate

Crosspoint array	*vin*	Number of outputs	V_{HL} (mV)	t_d (ns)	Fig.
1 × 2	1	1	598.801	4.065	Fig. 5b
1 × 64	1	63	528.881	6.223	Fig. 5c
1 × 2	0	1	−0.599	Not applicable	Fig. 5d
1 × 64	0	63	−35.559	Not applicable	Fig. 5e

vin denotes a logical input; V_{HL} is the voltage on horizontal nanowire HL, and t_d is the switching delay

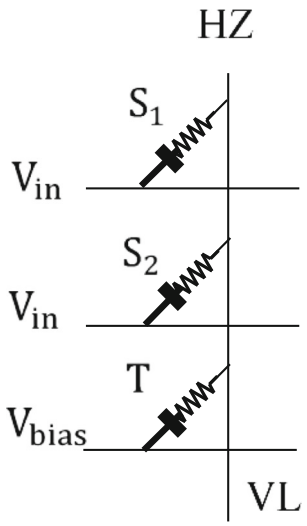


Fig. 6 A 3 × 1 crosspoint array used to perform two-input volistor NOR

3.4 Volistor AND Gate in a Crosspoint Array

The third basic gate in volistor logic is AND. This gate is realized as NOR with negated literals of the desired product applied to the crosspoint array. For example, the desired product $\bar{a}b\bar{c}$ is realized by applying $(vin_1, vin_2, vin_3) = (1, 0, 1)$ to the memristors. As a result, the voltage across T is sufficient to toggle t to "0" where $t = \overline{a + \bar{b} + c}$ and according to De Morgan's Law $\overline{a + \bar{b} + c} = \bar{a}b\bar{c}$ which is the desired result. The AND gate can be scaled to perform multi-input multi-output operations. The details are the same as described for volistor NOR. In this work, it is assumed that negated inputs are always available at the same cost as the non-negated inputs. Volistor logic allows a multi-input AND to be realized in only one pulse.

3.5 Volistor OR and NAND Gates in a Crosspoint Array

In all volistor logic gates discussed above, the logical coding scheme of memristive switch T is defined as follows: an open switch encodes logic "0" and a closed switch encodes logic "1." This scheme allows direct implementation of volistor NOR/AND; however, it does not allow for direct implementation of volistor OR/NAND. Therefore, executing a circuit with a volistor OR/NAND gate would require two consecutive pulses.

However, if the logical coding scheme of the memristor switch T is reversed, i.e., the open switch encodes logic "1" and the closed switch encodes logic "0," then the OR/NAND gate can be realized in one pulse. This scheme allows a direct realization of volistor OR in the same manner as volistor NOR described in Section 3.3. Likewise, volistor NAND is realized by applying negated input logic values to the crosspoint array in the same manner as the volistor AND described in Section 3.4. The designer could use one of the two encoding schemes for the entire system or use both encoding schemes in a partitioned system. In each separate partition, one encoding scheme is utilized and partitions with different schemes communicate through inverters.

Table 2 Implementation of multi-input single-output volistor NOR gate

Crosspoint array	Number of inputs		V_{VL}	t_d
	vin = 0	*vin* = 1	(mV)	(ns)
2 × 1	0	2	−599.400	4.048
	1	1	−598.203	4.068
	2	0	0.300	Not applicable
64 × 1	63	0	0.010	Not applicable
	0	63	−599.981	4.035
	50	13	−597.609	4.082

Parameters *vin*, V_{HL} and t_d are defined in the caption of Table 1

3.6 Mixed-Input Logic Gates in a Crosspoint Array

Inputs on standard volistor gates are voltages; however, as shown in Fig. 4d, implementing gates with mixed-inputs is possible using a hybrid of stateful and volistor logic where some inputs are represented by resistances and some by voltages. Figure 7 depicts a three-input NOR gate implemented in a 1 × 4 crosspoint array where $(s, vin_1, vin_2, t) = (0, 0, 1, 1)$. In (s, vin_1, vin_2, t), s and t are resistive logic values of S_1 and T, and vin_1 and vin_2 are logic input voltages applied to S_2 and S_3, see Fig. 7. Specifically, s is the resistive logic value stored in S_1 and it is to be interpreted as logic "0." Assume s has been set in a previous operation and that all memristors with voltage inputs have already been set to R_{CLOSED}. As in stateful logic, HL is grounded through R_G and V^+ is applied to S_1. As in volistor logic input data are applied to memristors S_2 and S_3, and bias voltage V^- is applied to memristor T. Equation (3), discussed in Section 3.1, requires $V_{bias} = V^-$ and $V_{in} \geq 0V$. With this configuration, the resistive value of logic s manifests as a voltage on HL and the circuit operates in the same manner as a

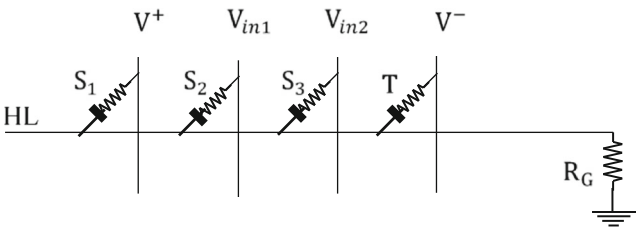


Fig. 7 Mixed-input NOR. The implementation of three-input one-output NOR gate. The resistive input is stored in S_1 and the voltage inputs are applied to S_2 and S_3. The output is stored in memristor T

volistor NOR. Translation between logical encoding schemes is accomplished in the same manner as described in Section 3.5. Mixed volistor AND gate is realized with one extra step; since all input resistances need to be negated, one additional pulse is required.

4 Hybrid Approach to Synthesize Boolean Functions in Crossbar Networks

In theory [10, 11], every single-output Boolean function of n variables can be realized in a crosspoint array with n source memristors and two additional working memristors. These working memristors can act as both source and target memristors. However, this realization leads to long sequences of pulses. These long sequences can be avoided by the use of crossbar arrays to realize arbitrary multi-output Boolean functions. Crosspoint arrays are the building blocks of crossbar arrays.

In this section, we first discuss hybrid computation in a crosspoint array and then we show how to implement arbitrary Boolean functions in a network of crossbar arrays. This network uses a combination of stateful, volistor, and mixed-input gates. These generic network structures can be used to implement logic functions in several forms such as SOP (Sum of Product), POS (Product of Sum), and TANT (Tree level AND NOT Network) [12]. TANT architecture requires non-inverted inputs, but we consider a generalized TANT with no restriction on input polarity. In this paper, stateful operations are realized solely with NOR and NOT. Note that the NOT gate can be obtained from INH by setting the non-inverted input to constant 1. Also, the NOR gate can be created by cascading stateful INH gates and setting the non-inverted input of the first gate of the cascade to constant 1, e.g., $\overline{a+b} = (1 \cdot \overline{a}) \cdot \overline{b}$.

4.1 Hybrid Computation in a Crosspoint Array

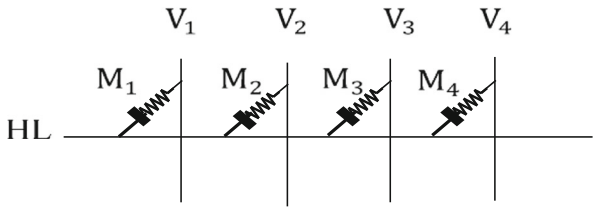
The simplest structure to perform memristive logic computation is a crosspoint array. In this structure, computations can be performed by two approaches: (1) solely stateful logic [1], (2) a combination of mixed-input, stateful and volistor logic. We call the second approach the hybrid approach. The first approach is potentially slow since a long sequence of operations needs to be implemented. However, the second approach has the potential to reduce the number of required operations. We assume a SOP with M single literal degenerate products and N products with more than one literal, thus the SOP is the OR of $M+N$ inputs. Realization of this SOP function requires a four-step process to be computed with the second approach:

(1) CLEAR: Set all memristors to the closed state. In a $1 \times n$ array, this requires driving all vertical nanowires to V^+ and the horizontal nanowire to V^-; for an $n \times 1$ array, the driving voltages should be swapped. This step is realized in one pulse.
(2) AND: Sequentially perform N volistor AND gates, each in one pulse. This step requires a total of N pulses.
(3) NOR: Perform a NOR operation on $N+M$ arguments. This includes the M single literal variables, which are supplied as voltages, and the N products from the previous step, which are stored as resistances. This step requires one pulse.
(4) NOT: Negate the result of the previous step. This step requires one pulse.

Example 1 The following example describes the hybrid approach in a crosspoint array for the Boolean function $f = ab + \overline{a}\overline{b} + c$. This function can be realized in a 1×4 crosspoint array in five consecutive operations, one for CLEAR, two for AND, one for NOR, and one for NOT, as shown in Fig. 8.

Figure 8a shows a 1×4 crosspoint array driven by V_i, and Fig. 8b shows the circuit configuration in each step. The first step is implemented by setting all V_i to V^+ and HL to V^-. This step will set the memristors M_i closed, i.e., encoding logic "1." The second step is to compute the product ab. This step is implemented by setting V_1 and V_2 to $V_{\bar{a}}$ and $V_{\bar{b}}$ where $V_{\bar{a}}$ and $V_{\bar{b}}$ are voltage signals encoding literals $\bar{a}$ and $\bar{b}$, respectively. The result of this computation is stored in M_4. Similarly, the second product, $\overline{a}\overline{b}$, is computed by setting V_1 and V_2 to V_a and V_b where V_a and V_b are voltage signals encoding literals a and b, respectively. The computed result is stored in M_3. The third step is realized with a mixed-input NOR gate, i.e., NOR $(ab,\ \overline{a}\overline{b},\ c)$. Note that variable c in SOP function f is a voltage signal, whereas ab and $\overline{a}\overline{b}$ are resistive signals. This step is implemented by connecting HL to the ground through R_G and driving V_1, V_2, V_3, and V_4 to V_c, V^-, V^+, and V^+, respectively. The result of the NOR gate is stored in M_2. Currently, the logic value of M_2 is $\overline{f}$, thus the last step is to invert this logic value to obtain f. This step is implemented by connecting HL to ground through R_G and driving V_1 and V_2 to V^- and V^+, respectively. Since M_3 and M_4 do not take part in the last computation, V_3 and V_4 are terminated to HZ.

As illustrated in Example 1, every SOP can be realized in $N+3$ operations, requiring at least $N+1+M$ memristors in the crosspoint array. In Example 1, $M=1$, so we used four memristors. As a matter of proper operation, all target memristors T_i must be initially closed. Further, if any of the inputs are logic "1," then at least one of those logic "1" inputs must be driving a closed source memristor S_i. If all such memristors S_i were open, the electrical characteristics of the memristors might prevent proper manifestation of the input



(a)

Operation	Step	Memristors' Drivers V_i					Logic state of Memristors. M_i			
		HL	V_1	V_2	V_3	V_4	M_1	M_2	M_3	M_4
1	CLEAR	V^-	V^+	V^+	V^+	V^+	1	1	1	1
2	AND	HZ	$V_{\bar{a}}$	$V_{\bar{b}}$	HZ	V^-	1	1	1	ab
3		HZ	V_a	V_b	V^-	HZ	1	1	$\bar{a}\bar{b}$	ab
4	NOR	R_G	V_c	V^-	V^+	V^+	1	$\bar{f}$	$\bar{a}\bar{b}$	ab
5	NOT	R_G	V^-	V^+	HZ	HZ	f	$\bar{f}$	$\bar{a}\bar{b}$	ab

(b)

Fig. 8 Example of hybrid computation in a crosspoint array. **a** A 1 × 4 crosspoint array used to implement SOP function *f*. **b** The circuit configuration to implement each step. The total number of consecutive operations (pulses) to realize *f* is 5

voltage on the common nanowire. Therefore, all memristors should be initially closed. The closed state of a memristor will only change to open when driven by V_{bias} and the voltage drop across the memristor is V_{OPEN}. Therefore, a target memristor T can be used as a source memristor, i.e., driven by V_{in} in subsequent operations with no risk of changing its state. This reuse allows for compact logic implementations. However, this approach potentially results in all source memristors being open, so any subsequent logic "1" inputs have no closed memristor to drive. Therefore, the number of memristors in the array should be more than the number of products, i.e., extra memristors should be provided as dedicated targets. So, realizing volistor gates with many inputs is possible neglecting the resistance of the common nanowire. This allows implementing structures such as SOP with many products. However, in a crosspoint array, each nanowire is driven by an individual CMOS driver with control circuitry, as shown in Fig. 2. This requirement imposes a large area penalty potentially restricting the number of inputs to the gates. Extending a crosspoint array to a crossbar array overcomes this potential limitation and increases the overall memristive density.

4.2 Hybrid Computation in a Crossbar Array

The crosspoint array can be scaled into a two-dimensional crossbar array of size $m \times n$. Each of the m rows in the crossbar array can be thought of as a $1 \times n$ crosspoint array and each of the n columns can be thought of as an $m \times 1$ crosspoint array. With this crossbar structure, in a single volistor operation, a product of literals can be created and copied to an arbitrary column memristor or row memristor in the crossbar simultaneously. The multi-output capability of crosspoint arrays discussed earlier allows an arbitrary number of copy operations to be performed in any of the two dimensions of the crossbar array. Figure 9a depicts target memristors as multi-input volistor gates whose outputs are the states of the corresponding memristors. Using the multi-output property, any combination of NOR/AND gates can be copied to any number of arbitrary locations in a single column. One operation is required for each gate type. In the worst case, the entire crossbar array can be populated with an arbitrary combination of gates in $2n$ operations. Populating the crossbar array in this manner is the first step (after initialization) to map a Boolean function to a generic crossbar fabric. This fabric may, in general, combine the use of volistor logic, mixed-input logic, and stateful logic in the same crossbar array to produce logic computations.

Example 2 As a means of discussing this hybrid approach, take for example, the implementation of function $f = \overline{\overline{a\bar{b} + \bar{c}\bar{d}} + \bar{A}} = \overline{\bar{B} + \bar{A}} = AB$. This implementation requires the same four-step process described in Section 4.1 and is illustrated in Fig. 10. The entries of the symbolic matrices represent the logic values stored in each memristor of the crossbar. Voltages applied to the horizontal and vertical nanowires are indicated by the values shown to the left of and on top of the matrices. The labels on top of the arrows between matrices indicate the number of operations required to move to the next matrix. Figure 10a shows the crossbar after initialization; the voltages to achieve this state are not shown. Figure 10b shows the result of computing the first product with volistor AND. Figure 10c shows the computation of the other product which is also realized with volistor AND. Figure 10d shows stateful logic being used to compute

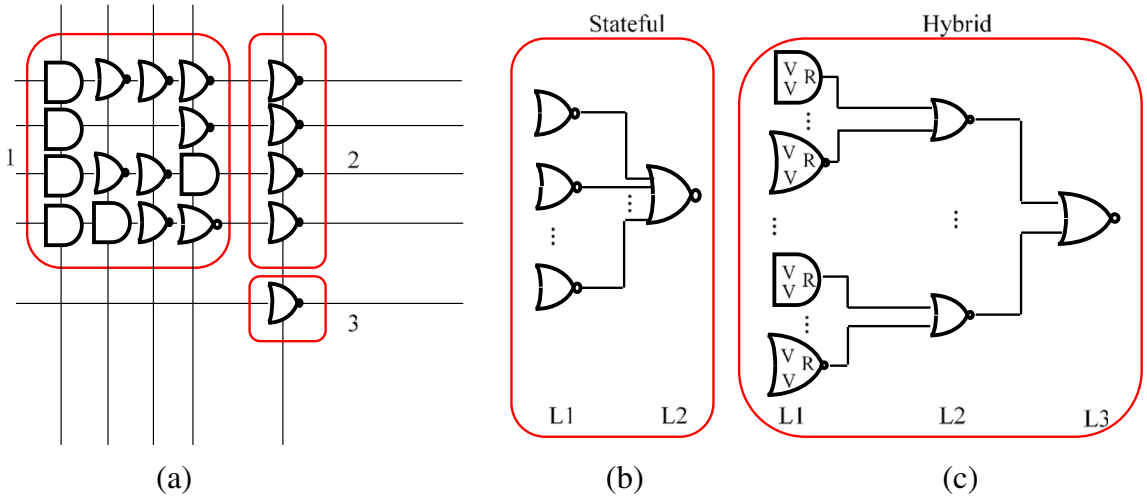


Fig. 9 The crossbar array. **a** A crossbar array divided into three blocks; the populated gates in the block labeled *1* are realized by volistors, but the blocks labeled *2* and *3* are realized with stateful NOR. **b** The symbolic two-level circuit is realized with the stateful approach. **c** The symbolic three-level circuit is realized with the hybrid approach. The first level of the circuit, *L1*, is realized with volistor logic, whereas the next levels, *L2* and *L3*, are realized with the stateful approach

$\overline{B} = \overline{a\bar{b} + \bar{c}\bar{d}}$. In this step, V^+ and V^- are equal to V_{COND} and V_{SET}, and R_G is the load resistor in stateful INH [2]. Figure 10e shows the implementation of a mixed NOR gate with resistance-based signal $\overline{B}$ and voltage-based signal $\bar{A}$ producing the desired function *f*.

With the hybrid approach, the use of stateful IMP can be avoided. This simplifies the driver circuitry, removes the need for a keeper circuit [13], and simplifies crossbar initialization since all memristors are initialized as closed. The stateful approach also requires a step to store the inputs in the crossbar. However, since the hybrid approach uses voltages as inputs, no storage step is required. The main advantage of the hybrid approach over the stateful approach is that, for the same number of operations and memristors, one additional logic level can be realized. The hybrid approach uses the efficiency of volistors to implement the first level of logic which is usually most complex. Figure 9a depicts a crossbar array divided into three blocks. In stateful logic, the memristors in block 1 store inputs; the memristors in block 2 store the first-level logic outputs, and the memristor in block 3 stores the output of the second-level logic. In the hybrid approach, since inputs are voltages, the memristors in block 1 store the outputs of the first-level logic; the memristors in block 2 store the outputs of the second-level logic, and the memristor in block 3 stores the output of the third-level logic. With the same number of operations and the same number of memristors, the stateful approach produces only two-level logic (Fig. 9b) whereas the hybrid approach produces three-level logic as shown in Fig. 9c. However, since in volistor logic the inputs are voltages, there can be only one set of inputs applied to the crossbar at any given moment and therefore only one output can be computed at a time. In stateful logic, inputs are stored as

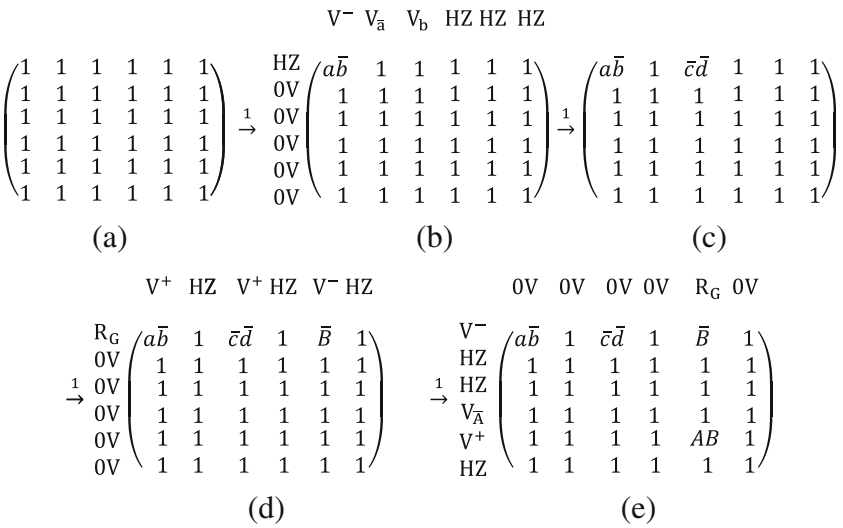


Fig. 10 The symbolic matrices illustrate the steps of logic computations based on the hybrid approach for computing $f = \overline{\overline{a\bar{b} + \bar{c}\bar{d}} + \bar{A}}$. **a** Initialization step. **b** Computing $a\bar{b}$ with a volistor AND. **c** Computing $\bar{c}\bar{d}$ with a volistor AND. **d** Computing $\overline{a\bar{b} + \bar{c}\bar{d}}$ with a stateful NOR. **e** Computing *f* with a mixed-gate NOR

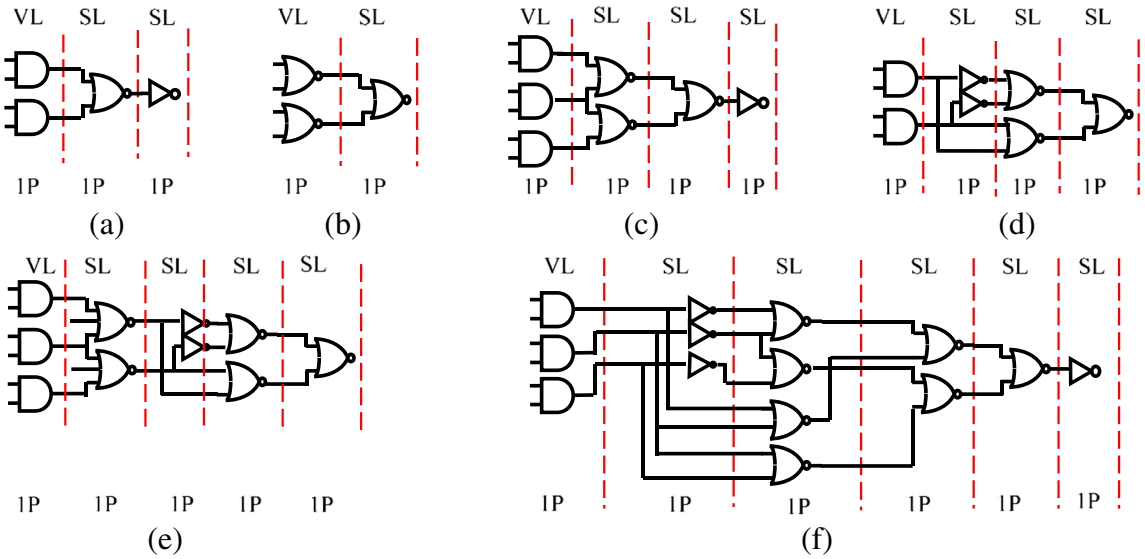


Fig. 11 Implementation of different forms of Boolean functions with the hybrid approach. **a** An example of a SOP function. **b** An example of a POS function. **c** An example of a three-level sum of products of sums. **d** An example of an EXOR of two products. **e** An example of a NAND-AND-EXOR logic function. **f** An example of an AND-EXOR-OR logic function. *P* stands for pulse (operation), e.g., *1P* indicates that a related logic level requires one pulse to be implemented. *VL* and *SL* stand for volistor and stateful operations. In all circuits, only the first logic level is implemented with volistor logic

resistances. Therefore, each row or column can be thought of as a distinct set of inputs capable of simultaneously producing distinct outputs, with the restriction that all outputs implement the same type of stateful gate [2], e.g., all the four NOR gates as shown in block 2 of Fig. 9a. In Section 4.3, we use the hybrid approach in a crossbar network to achieve different outputs in parallel.

4.3 Hybrid Computation in a Crossbar Network

In a data path of a larger memristor architecture, there are usually several combinational blocks as well as memories each of which is realized in one or more crossbars. A question arises as to how one can implement information transfer between these crossbars. Let us consider transfer of data from source memory to target combinational logic. The memories can store information as either voltage (e.g., CMOS memory) or as resistance. If data are stored as voltage, the first level of the target logic can be implemented with volistors. However, when data in memories are stored as resistances, the first logic level in combinational logic is realized with stateful NOR/NOT. Another problem is how to partition large combinational blocks to improve processing time by increasing parallelism. In a single crossbar array using volistor logic, identical gates with identical inputs can be produced in a single step. This is a limited form of parallelism that replaces fan-out. This limitation can be resolved using a network of many individual crossbars. Each individual crossbar can have as many copies as necessary of a single gate with identical inputs. This structure allows two or more separate crossbars to simultaneously calculate the first logic level of an arbitrary Boolean function. Additional logic levels can be achieved with stateful operations, mixed-input gates, or both. Table 3 shows different structures to realize Boolean functions that are well suited to this hybrid approach. Next, we present detailed examples for cases 2 and 5 from Table 3.

Example 3 POS implementation. The three variable EXOR expression $f = a \oplus b \oplus c$ can be implemented as the POS expression $(a+b+c)(a+\overline{b}+\overline{c})(\overline{a}+\overline{b}+c)(\overline{a}+b+\overline{c})$. The De Morgan equivalent expression is $f = \overline{(a+b+c)} + \overline{(a+\overline{b}+\overline{c})} + \overline{(\overline{a}+\overline{b}+c)} + \overline{(\overline{a}+b+\overline{c})} = \overline{w+x+y+z}$. The function is synthesized in four crossbar arrays each of which is 4×4. In this crossbar column, adjacent crossbars communicate through vertical switches capable of connecting and disconnecting the crossbars as illustrated in Fig. 12a. Figure 12b shows the crossbar column with closed vertical switches between all adjacent crossbars creating a 16×4 crossbar array. These switches are not further discussed in this work. Executing f is a three-step procedure.

(1) Initialize all crossbar arrays by driving all vertical nanowires with V^+ and all horizontal nanowires with V^-.

(2) Simultaneously compute volistor NOR of each maxterm in a separate crossbar column as described in Section 3 3.3. In Fig. 12a, individual crossbar arrays A, B, C, and D compute $w = \overline{a+b+c}$, $x = \overline{a+\overline{b}+\overline{c}}$, $y = \overline{\overline{a}+\overline{b}+c}$, and $z = \overline{\overline{a}+b+\overline{c}}$ in parallel.

Table 3 Different forms of logic functions and their De Morgan's equivalents

#	Logical form	De Morgan's equivalent	Example	# of pulses
1	AND-OR (SOP)	AND-NOR-NOT	Fig. 11a	3
2	OR-AND (POS)	NOR-NOR	Fig. 11b	2
3	NAND-NAND-NAND	AND-NOR-NOR-NOT	Fig. 11c	4
4	AND-EXOR (ESOP)	AND-NOT-NOR-NOR	Fig. 11d	4
5	NAND-AND-EXOR	AND-NOR-NOT-NOR-NOR	Fig. 11e	5
6	AND-EXOR-OR	AND-NOT-NOR-NOR-NOR-NOT	Fig. 11f	7

(3) Connect the vertical switches between adjacent crossbars to form a 16 × 4 crossbar array. Perform a stateful NOR on the logic values computed in step 2 as illustrated in Fig. 12b.

Executing f with a solely stateful approach would require two steps plus the overhead of setting up each input [2]. This input overhead is directly proportional to the number of inputs in the largest sum. Volistor logic has no such input overhead; a multi-input gate and a single-input gate are both produced in the same number of steps. However, the solely stateful approach can be executed in two steps in a single crossbar whereas the hybrid approach would require more steps if performed in a single crossbar.

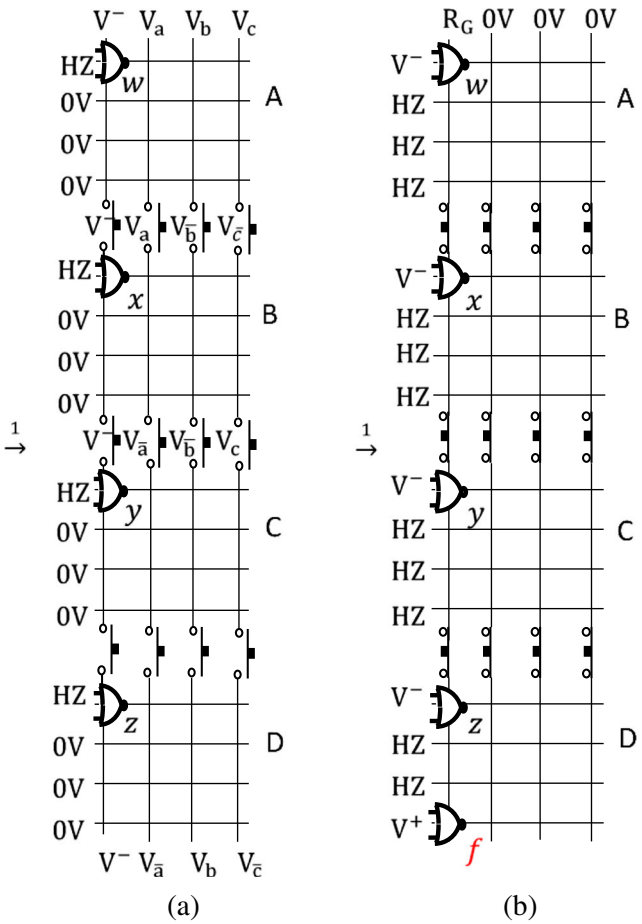


Fig. 12 Implementation of POS function f in a crossbar network comprised of four crossbar arrays of size 4 × 4. The function is realized in a two-step procedure. **a** Realization of the first logic level of POS function f in separate crossbars. The network configuration shows the voltage drivers applied to each nanowire. This step produces four NOR gates. **b** The switches between the 4 × 4 arrays are closed to create a 16 × 4 crossbar array. The results w, x, y, and z of the first step are NORed to create the output of the POS function f

Example 4 NAND-AND-EXOR implementation. Consider the expression $f = e\cdot\overline{a}b\cdot\overline{c}d \oplus \overline{c}d\cdot\overline{a}e\cdot b$. The De Morgan equivalent expression is $f = \overline{\overline{e} + \overline{a}b + \overline{c}d + ae + \overline{b}} + \overline{\overline{e} + \overline{a}e + \overline{c}d + \overline{\overline{c}d} + ae + \overline{b}} = \overline{\overline{e} + x + y + z + \overline{b}} + \overline{\overline{e} + x + y} + \overline{y + z + \overline{b}} = \overline{\overline{w1} + \overline{w2}} + \overline{w1 + w2} = \overline{o1 + o2}$. Executing f is the six-step procedure depicted in Fig. 13 and described as follows. In each step, all outputs are computed simultaneously.

(1) Perform the initialization as in example 3.
(2) Compute the products x, y, and z in parallel. Value y is computed twice to enable simultaneous computation in step 3 as in Fig. 13a. This realizes the four AND gates using volistors.
(3) Using mixed-input gates, as described in Section 3.6, compute $w1$ and $w2$ where $V_{\bar{e}}$ and $V_{\bar{b}}$ are input voltages and x, y, and z are input resistances, as in Fig. 13b. This realizes the NOR logic level.
(4) Produce $\overline{w1}$ and $\overline{w2}$ in parallel, as in Fig. 13c. This performs two stateful NOT operations in the third logic level.
(5) Produce $o2$ and $o1$ in the fourth logic level, NOR, using the outputs calculated in steps 3 and 4, as in Fig. 13d.
(6) Produce f from $o2$ and $o1$, as in Fig. 13e. This produces the fifth level of logic, NOR. Executing f with a solely stateful approach would require six steps plus the input setup overhead. Note that NAND-AND-EXOR circuits are a new concept in logic synthesis. These circuits are a generalization of the PSE circuits introduced in [14] in which only one argument of the AND layer is a NAND and others are literals.

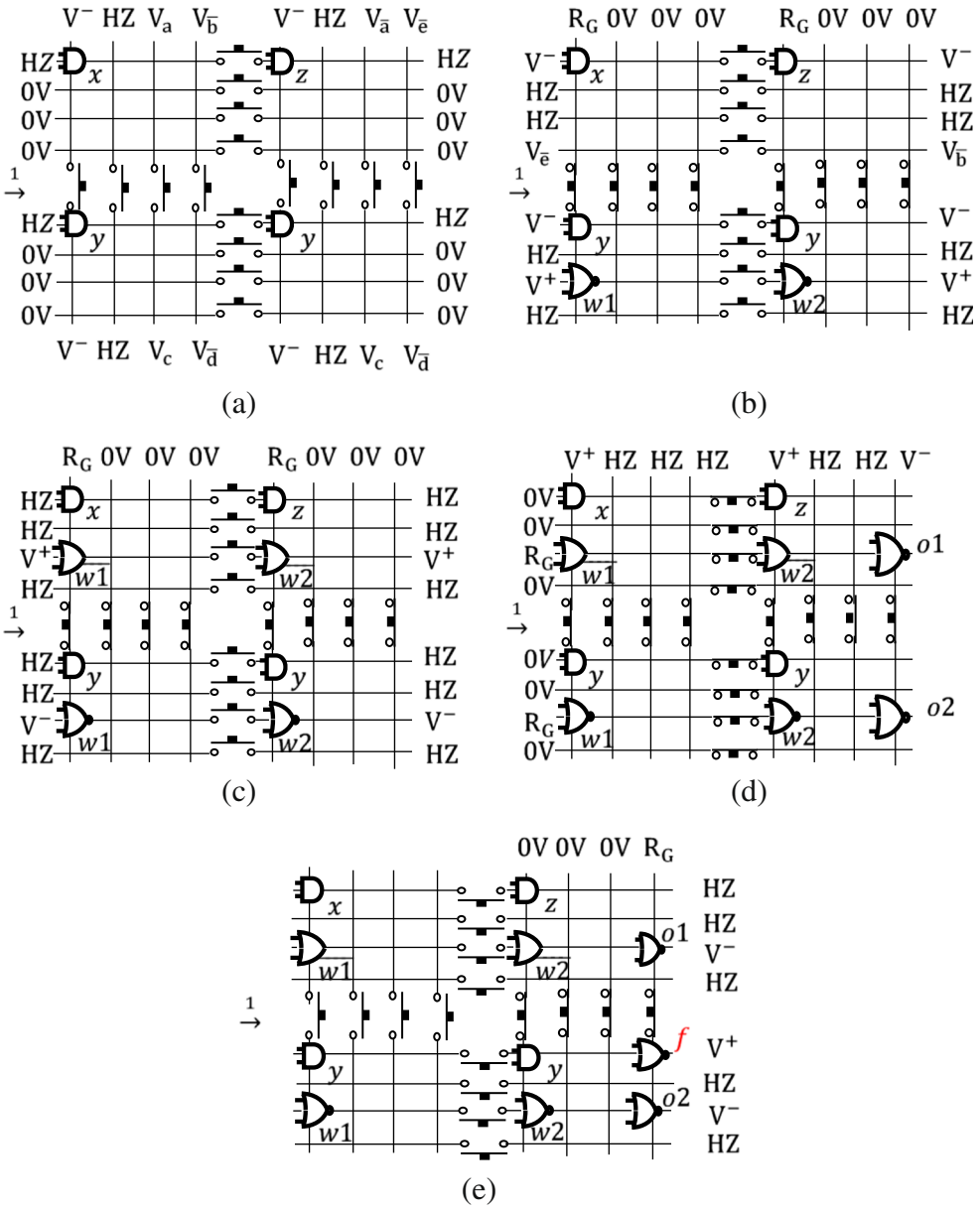


Fig. 13 Realization of the De Morgan equivalent for the NAND-AND-EXOR expression *f*. The crossbar network is comprised of four crossbar arrays each of size 4 × 4. The function is realized in a six-step procedure. **a** The first logic level of *f* is realized with volistors, **b** the second logic level of *f* is realized with mixed-input gates, and **c**–**e** the rest of the logic levels are realized with stateful logic. Crossbar drivers are indicated by 0 V, V^+, V^-, R_G, and HZ. In each step, the performed operation is depicted by logic gates and the results of the operations are shown as outputs of the gates

5 Comparison of Power Consumption and Speed Between Stateful and Volistor Circuit Models

In this work, power measurements are made using the same model mentioned in Section 2 in the LTspice IV simulation environment. The average power P_{avg} consumed in a crosspoint array is computed using the equation $P_{avg} = \sum V_{RMS} \cdot I_{RMS}$ where V_{RMS} and I_{RMS} are the root mean squares of the voltage across and the current through each memristor. The average power consumption of each individual element of a $1 \times n$ crosspoint array where $n \leq 64$ is computed over a 10 ns interval beginning with the application of the driver voltages $V^+ = 0.6V$, $V^- = -0.6V$ and 0 V which are applied for 8 ns. Let the power consumed in a source memristor set to logic "1" be denoted P_{S1}, the power consumed in a source memristor set to logic "0" be denoted P_{S0}, the power consumed in a target memristor be denoted P_T, and the power consumed in load resistor R_G be denoted P_{RG}. Let P_S denotes the total power consumption in all source memristors. The superscripts SL and VL are used to indicate power consumptions during stateful logic and volistor logic, respectively. For example, P_{S0}^{SL} is the power consumed by a source memristor set to logic "0" when SL is performed and P_S^{VL} is the power consumed in all source memristors when volistor logic is performed. Table 4 describes the power consumption in each crosspoint element where V_{HL} and $\hat{V}_{HL}$ are the voltages on the horizontal nanowire during VL and SL operations, respectively. Table 4 is derived based on the average power equation P_{avg} and Ohm's Law; the circuit elements

Table 4 Power consumption in each crosspoint array element

Logical op.	P_{S0}	P_{S1}	P_T	P_{RG}
VL	$\frac{(V_{HL})^2}{R_{OPEN}}$	$\frac{(V^+-V_{HL})^2}{R_{OPEN}\times 10^{-3}}$	$\frac{(V^++V_{HL})^2}{R_{OPEN}}$	0
SL	$\frac{(V^+-\hat{V}_{HL})^2}{R_{OPEN}}$	$\frac{(V^+-\hat{V}_{HL})^2}{R_{OPEN}\times 10^{-3}}$	$\frac{(V^++\hat{V}_{HL})^2}{R_{OPEN}}$	$\frac{\hat{V}_{HL}^{\ 2}}{R_{OPEN}}\times 10\sqrt{10}$

P_{S0} and P_{S1} denote the power consumptions in a source memristor set to logic "0" and logic "1", respectively. P_T is the power consumption in a target memristor, and P_{RG} is the power consumption in load resistor R_G

considered in power computation are the resistance values of the memristors, either R_{CLOSED} or R_{OPEN}, load resistance R_G, and the driver voltages. The power required to connect the nanowires to V^+, V^-, 0V, R_G, and HZ is the same in both VL and SL operations and is not considered in this work. The resistance of the horizontal nanowire HL is negligible when compared with load resistance R_G. Jo et al. [15] reported that the resistive value of a relatively large width nanowire of diameter 120 nm used in a 1-kb crossbar array is at most 30 KΩ when implemented with relatively high resistance p-doped Si. This upper bound is much smaller than $R_G = 15$ MΩ chosen in this work. Recall, $R_G = \sqrt{R_{OPEN} \times R_{CLOSED}}$ and $\frac{R_{OPEN}}{R_{CLOSED}} = 10^3$ as mentioned in Section 2 and Section 3. The power consumed by volistor logic in a crosspoint array is entirely due to the leakage through the reverse biased rectifying memristors as there is no direct path to ground. However, when volistor logic is performed in a crossbar array, there is an additional power consumption in memristors not taking part in the operations. This is true for stateful logic as well. Since the power consumption in both cases is the same and we are only interested in comparing SL and VL, this additional power is not discussed. In this section, the power consumption in a crosspoint array for $S_1 > 0$ and $S_1 = 0$ is analyzed separately. For ease of reference, let S_1 and S_0 denote the numbers of source memristors set to $vin = 1$ and $vin = 0$, respectively; T denotes the number of target memristors; P_{SL} and P_{VL} denote the total power consumption in memristors and in load resistor R_G during SL and VL operations, respectively; and t_d denotes the switching delay—the time required to completely switch from the close state of a target memristor T to the open state.

5.1 Analysis of Power Consumption and Switching Delays in a 1 × 8 Crosspoint Array for $S_1 > 0$

Table 5 compares the switching delay, t_d, and the overall power consumption in memristors and in load resistor R_G during SL and VL operations for various compositions of S_1, S_0 and $T, \frac{P_{SL}}{P_{VL}}$. From the simulation results shown in Table 5, the following is observed.

(1) In SL and VL, P_T is approximately 2nW, i.e., the changes to the number and the high/low composition of input values slightly affect the power consumed by any individual target memristor T. This is expected given the characteristics of the rectifying memristors and considering that $V_{HL} \approx V_{in}$.

(2) The major contributor to the power consumption in SL is R_G—it consumes approximately eight times more power than each target memristor. In VL, $P_{RG} = 0$.

(3) The minor contributors to the power consumption in SL is $S_0 \times P_{S0}^{SL}$.

Table 5 Circuit element power consumption, switching delay and power ratio in a 1 × 8 crosspoint array

(S_1, S_0, T)	Op	P_{S1} (nW)	P_{S0} (nW)	P_T (nW)	P_{RG} (nW)	t_d (ns)	P_{avg} (nW)	$\frac{P_{SL}}{P_{VL}}$
(2, 2, 1)	VL	2.224e-3	0.555	2.226	0	4.044	3.340	6.13
	SL	0.168	0.168e-3	2.192	17.946	4.239	20.474	
(2, 3, 1)	VL	3.471e-3	0.555	2.225	0	4.044	3.897	5.254
	SL	0.168	0.168e-3	2.192	17.947	4.239	20.476	
(2, 4, 1)	VL	4.993e-3	0.554	2.224	0	4.054	4.450	4.601
	SL	0.168	0.168e-3	2.192	17.947	4.239	20.476	
(2, 4, 2)	VL	8.868e-3	0.553	2.221	0	4.064	6.672	3.397
	SL	0.187	0.187e-3	2.190	17.911	4.227	22.666	
(2, 3, 3)	VL	11.22e-3	0.553	2.220	0	4.073	8.341	2.955
	SL	0.207	0.207e-3	2.187	17.876	4.288	24.645	
(2, 2, 4)	VL	0.014	0.552	2.219	0	4.083	10.008	2.701
	SL	0.229	0.229e-3	2.185	17.838	4.288	27.036	

S_0 and S_1 denote the numbers of source memristors set to logic "0" and logic "1", respectively; T denotes the number of target memristors. P_{S0} and P_{S1} are the power consumptions in a source memristor set to logic "0" and logic "1", respectively. P_T is the power consumption in a target memristor, and P_{RG} is the power consumption in load resistor R_G. P_{avg} denotes the average power consumption in the crosspoint array. t_d denotes the switching delay. P_{SL} and P_{VL} indicate the power consumptions during SL and VL operations, respectively

(4) The major contributors to the power consumption in VL are determined based on S_0 and T.
(5) For any combination of S_1, S_0, and T, $\frac{P_{SL}}{P_{VL}} > 2$. The lower bound of $\frac{P_{SL}}{P_{VL}}$ obtained for the composition of (S_1, S_0, T) = (1, 0, 7) *is* 2.129.
(6) For any combination of S_1, S_0, and T, t_d is always lower for VL.

As a summary, we observed that in a 1 × 8 crosspoint array VL circuit is faster than SL and consumes at most half the power of its SL technology equivalent.

In a 1 × 8 crosspoint array, for any composition of S_1, S_0, and T where $S_1 > 0$, the power ratio can be approximated based on the following power properties, Property 1–Property 4.

Property 1 When $S_1 > 0$, the power consumption in a target memristor during VL operation can be approximated as shown in (5).

$$P_T^{VL} \approx 4 \times P_{S0}^{VL} \tag{5}$$

Proof The power consumption in a target memristor during VL operation is $P_T^{VL} = \frac{(V_{HL}-V^-)^2}{R_{OPEN}} = \frac{(V_{HL}+V^+)^2}{R_{OPEN}}$ and the power consumption in a source memristor set to logic "0" during VL operation is $P_{S0}^{VL} = \frac{V_{HL}{}^2}{R_{OPEN}}$. So, $\frac{P_T^{VL}}{P_{S0}^{VL}} = \frac{(V_{HL}+V^+)^2}{V_{HL}{}^2}$. Let $V^+ = V_{HL} + \varepsilon$. As a result, $\frac{P_T^{VL}}{P_{S0}^{VL}} = \left[\frac{2V_{HL}+\varepsilon}{V_{HL}}\right]^2 = \left[2 + \frac{\varepsilon}{V_{HL}}\right]^2$. The upper and lower bounds of $\frac{P_T^{VL}}{P_{S0}^{VL}}$, obtained for the upper and lower bounds of $\frac{\varepsilon}{V_{HL}}$, are calculated for ($S_1$, S_0, T) = (1, 1 , 6) and (S_1, S_0, T) = (6, 1, 1), respectively. These compositions were simulated using LTspice obtaining $4.002 \le \frac{P_T^{VL}}{P_{S0}^{VL}} \le 4.052$. Therefore, for any combination of S_1, S_0, and T where $S_1 > 0$ we can assume $P_T^{VL} \approx 4 \times P_{S0}^{VL}$.

Property 2 When $S_1 > 0$, the power consumption in the load resistor can be approximated as shown in (6).

$$P_{RG} \approx 8 \times P_T^{SL} \tag{6}$$

Proof The power consumption in the load resistor is $P_{RG} = \frac{\hat{V}_{HL}{}^2}{R_G}$ and the power consumption in a target memristor during SL operation is $P_T^{SL} = \frac{(\hat{V}_{HL}-V^-)^2}{R_{OPEN}} = \frac{(\hat{V}_{HL}+V^+)^2}{R_{OPEN}}$. So, $\frac{P_{RG}}{P_T^{SL}} = \frac{\hat{V}_{HL}{}^2}{R_G} \times \frac{R_{OPEN}}{(\hat{V}_{HL}+V^+)^2} = \frac{R_{OPEN}}{R_G} \times \frac{1}{\left[\frac{\hat{V}_{HL}+V^+}{\hat{V}_{HL}}\right]^2}$. Let $V^+ = \hat{V}_{HL} + \hat{\varepsilon}$. As a result, $\frac{P_{RG}}{P_T^{SL}} = \frac{R_{OPEN}}{R_G} \times \frac{1}{\left[\frac{2\hat{V}_{HL}+\hat{\varepsilon}}{\hat{V}_{HL}}\right]^2} = \frac{R_{OPEN}}{R_G} \times \frac{1}{\left[2+\frac{\hat{\varepsilon}}{\hat{V}_{HL}}\right]^2}$. Computing the upper bound of $\frac{P_{RG}}{P_T^{SL}}$ requires computing the lower bound of voltage ratio $\frac{\hat{\varepsilon}}{\hat{V}_{HL}}$. The upper and lower bounds of $\frac{P_{RG}}{P_T^{SL}}$ or the lower and upper bounds of $\frac{\hat{\varepsilon}}{\hat{V}_{HL}}$ are calculated for the compositions of (S_1, S_0, T) = (7, 0, 1) and (S_1, S_0, T) = (1, 0, 7), respectively. These compositions were simulated using LTspice obtaining $7.542 \le \frac{P_{RG}}{P_T^{SL}} \le 7.866$. Therefore, for any combination of S_1, S_0, and T where $S_1 > 0$ we can assume $P_{RG} \approx 8 \times P_T^{SL}$.

Since P_T^{VL} and P_T^{SL} for any combination of S_1, S_0, and T where $S_1 > 0$ are approximately 2nW, they can be replaced with P_T in (5) and (6).

Property 3 When $S_1 > 0$, the power consumption in source memristors during SL operation is considerably smaller than the overall power consumption in target memristors and load resistor R_G as shown in (7).

$$\frac{P_S^{SL}}{P_{SL} - P_S^{SL}} \ll 1 \tag{7}$$

Proof The simulation results show that the power consumption in source memristors during SL operation, P_S^{SL}, is negligible when compared to the overall power consumption in target memristors and load resistor R_G. The power consumption in source memristors set to logic "1" is much larger than the power consumption in source memristors set to logic "0", as can be seen in follows: $\frac{P_{S1}^{SL}}{P_{S0}^{SL}} = \frac{\left(V^+ - \hat{V}_{HL}\right)^2}{R_{CLOSED}} \times \frac{R_{OPEN}}{\left(V^+ - \hat{V}_{HL}\right)^2} = 10^3$. Substituting $P_{S1}^{SL} = 10^3 \times P_{S0}^{SL}$ in $\frac{P_S^{SL}}{P_{SL} - P_S^{SL}}$ results in $\frac{(S_1 \times 10^3 + S_0) \times P_{S0}^{SL}}{P_{SL} - P_S^{SL}}$. Substituting $P_{SL} - P_S^{SL}$ with $T \times P_T + P_{RG}$ or its equivalent $(T+8) \times P_T$ in $\frac{(S_1 \times 10^3 + S_0) \times P_{S0}^{SL}}{P_{SL} - P_S^{SL}}$ results in $\frac{(S_1 \times 10^3 + S_0)}{(T+8) \times \frac{P_T}{P_{S0}^{SL}}}$. It is required to compute the lower bound of $\frac{P_T}{P_{S0}^{SL}}$ to verify $\frac{(S_1 \times 10^3 + S_0)}{(T+8) \times \frac{P_T}{P_{S0}^{SL}}} \ll 1$. This power ratio, $\frac{P_T}{P_{S0}^{SL}}$, is equal to $\frac{\left(\hat{V}_{HL} - V^-\right)^2}{R_{OPEN}} \times \frac{R_{OPEN}}{\left(V^+ - \hat{V}_{HL}\right)^2}$ or $\left(\frac{2\hat{V}_{HL}}{\hat{\varepsilon}} + 1\right)^2$ considering $V^+ = \hat{V}_{HL} + \hat{\varepsilon}$. The lower bound of $\frac{P_T}{P_{S0}^{SL}}$ obtained for the lower bound of $\frac{\hat{V}_{HL}}{\hat{\varepsilon}}$ is calculated for (S_1, S_0, T) = (1, 1, 6). This composition results in $\frac{\hat{V}_{HL}}{\hat{\varepsilon}} = 21.949$ and therefore $\frac{(S_1 \times 10^3 + S_0)}{(T+8) \times \frac{P_T}{P_{S0}^{SL}}} = 0.036$ which is considerably smaller than 1 and thus proves (7). The immediate consequence of (7) for any combination of S_1, S_0 and T where $S_1 > 0$ is $P_{SL} - P_S^{SL} \approx P_{SL}$ used in deriving (9).

Property 4 The power consumption in source memristors driven by logic "1" during VL operation is considerably smaller than the overall power consumption in source

memristors driven by logic "0" and target memristors as in (8).

$$\frac{S_1 \times P_{S1}^{VL}}{P_{VL} - S_1 \times P_{S1}^{VL}} \ll 1 \tag{8}$$

Proof The simulation results show that the power consumption in source memristors driven by logic "1" during VL operation is negligible when compared to the overall power consumption in memristors. Using the results of simulations, inequality (8) can be proved as follows. $\frac{S_1 \times P_{S1}^{VL}}{P_{VL} - S_1 \times P_{S1}^{VL}} = \frac{S_1 \times P_{S1}^{VL}}{S_0 \times P_{S0}^{VL} + T \times P_T}$. Using (6), $\frac{S_1 \times P_{S1}^{VL}}{S_0 \times P_{S0}^{VL} + T \times P_T} \approx \frac{S_1 \times P_{S1}^{VL}}{S_0 \times \frac{P_T}{4} + T \times P_T} = \frac{S_1}{\frac{P_T}{P_{S1}^{VL}}\left(\frac{S_0}{4} + T\right)}$. The lower bound of $\frac{P_T}{P_{S1}^{VL}}$ is required to verify $\frac{S_1 \times P_{S1}^{VL}}{P_{VL} - S_1 \times P_{S1}^{VL}} \ll 1$ and is obtained for the composition of $(S_1, S_0, T) = (1, 0, 7)$. Note that $\frac{P_T}{P_{S1}^{VL}} = \frac{(2V_{HL} + \varepsilon)^2}{R_{OPEN}} \times \frac{R_{CLOSED}}{(V^+ - V_{HL})^2} = 10^{-3} \times \left(\frac{2V_{HL}}{\varepsilon} + 1\right)^2$ where $\varepsilon = V^+ - V_{HL}$. Using LTspice simulator, the lower bound of $\frac{V_{HL}}{\varepsilon}$ for the given combination is 70.929 and consequently $\frac{P_T}{P_{S1}^{VL}} = 20.409$. Therefore, $\frac{S_1 \times P_{S1}^{VL}}{P_{VL} - S_1 \times P_{S1}^{VL}} \approx 0.007$ is significantly smaller than 1 and thus proves (8). The immediate consequence of (8) for any combination of S_1, S_0, and T where $S_1 > 0$ is $P_{VL} - S_1 \times P_{S1}^{VL} \approx P_{VL}$ used in deriving (9).

According to (7) and (8), $\frac{P_{SL}}{P_{VL}} \approx \frac{T \times P_T + P_{RG}}{S_0 \times P_{S0}^{VL} + T \times P_T}$, and according to (5) and (6), $\frac{T \times P_T + P_{RG}}{S_0 \times P_{S0}^{VL} + T \times P_T} \approx \frac{T \times P_T + 8 \times P_T}{S_0 \times \frac{P_T}{4} + T \times P_T}$. Therefore, for any combination of S_1, S_0, and T where $S_1 > 0$, we can assume:

$$\frac{P_{SL}}{P_{VL}} \approx \frac{T + 8}{T + \frac{S_0}{4}} \tag{9}$$

The lower bound of (9) is larger than 2. Simplifying the inequality $\frac{T+8}{T+\frac{S_0}{4}} > 2$ results in $T + \frac{S_0}{2} < 8$. In a 1×8 crosspoint array for any composition of S_1, S_0, T where $S_1 > 0$, the inequality $T + \frac{S_0}{2} < 8$ holds and so $\frac{P_{SL}}{P_{VL}} > 2$.

In terms of switching delay, volistor gates perform faster than stateful gates. The switching delay in both SL and VL operations corresponds to the voltage drop across the target memristors, or equivalently the voltage of the common nanowire during the operations. The lowest voltage drop across the target memristors is obtained for the composition of (S_1, S_0, T) = (1, 0, 7), for which t_d is 4.136 and 4.477 ns for VL and SL, respectively. The highest voltage drop across the target memristors is obtained for the composition of (S_1, S_0, T) = (7, 0, 1) for which t_d is 4.030 and 4.089 ns for VL and SL, respectively. Note that the increase of S_1 reduces the switching delay in both SL and VL operations.

5.2 Analysis of Power Consumption in a $1 \times n$ Crosspoint Array for $S_1 = 0$

In a 1×8 crosspoint array, for $S_1 = 0$ and any combination of S_0 and T, our simulation results show that $\frac{P_{SL}}{P_{VL}} > 1$. Table 6 compares the power consumption in memristors and load resistor R_G for $S_1 = 0$ and various compositions of S_0 and T during VL and SL operations. In all compositions, T is kept constant at 1 and S_0 is varied between 1 and 7. Table 6 shows that the increase of S_0 has almost no effect on P_{VL}, i.e. for all given compositions the power consumption in memristors during VL operation is almost equal to the power consumption in only target memristors.

Property 5 When $S_1 = 0$, the power consumption in source memristors driven by logic "0" during VL operation is considerably smaller than the overall power consumption in target memristors as in (10).

$$\frac{S_0 \times P_{S0}^{VL}}{T \times P_T^{VL}} \ll 1 \tag{10}$$

Proof Our simulation results show that the overall power consumption in memristors during VL operation is almost equal to the power consumption in only target memristors when $S_1 = 0$.

Table 6 Comparison of power ratio $\frac{P_{SL}}{P_{VL}}$ of a multi-input NOR realized in a 1×8 cross point array with $S_1 = 0$ and various values of S_0

(S_1, S_0, T)	Op.	P_{S0} (nW)	P_T (nW)	P_{RG} (nW)	P_{avg} (nW)	$\frac{P_{SL}}{P_{VL}}$
(0, 1, 1)	VL	0.556e-3	0.556	0	0.557	2.004
	SL	0.558	0.558	0	1.116	
(0, 2, 1)	VL	0.139e-3	0.557	0	0.557	2.975
	SL	0.527	0.589	0.014	1.657	
(0, 4, 1)	VL	0.035e-3	0.557	0	0.557	4.772
	SL	0.474	0.648	0.114	2.658	
(0, 7, 1)	VL	0.011e-3	0.557	0	0.557	6.427
	SL	0.407	0.731	0.392	3.58	

The parameters S_0, S_1, T and so on are defined in the caption of Table 5

The power ratio $\frac{S_0 \times P_{S0}^{VL}}{T \times P_T^{VL}}$ is equal to $\frac{S_0}{T} \times \frac{V_{HL}{}^2}{R_{CLOSED}} \times \frac{R_{OPEN}}{(V_{HL}+V^+)^2} = 10^3 \left(\frac{S_0}{T}\right) \times \frac{1}{\left(1+\frac{V^+}{V_{HL}}\right)^2}$. The upper bound of $\frac{P_{S0}^{VL}}{P_T^{VL}}$ is obtained for the composition of $(S_0, T) = (7, 1)$ which produces maximum V_{HL}. For this composition, $V_{HL} = -75.417\mu V$ and thus $\frac{P_{S0}^{VL}}{P_T^{VL}} = 1.106 \times 10^{-4}$ which is considerably smaller than 1 and proves (10). Note that for a larger crosspoint array equation (10) is also valid, e.g., in a 1×64 crosspoint array the upper bound of $\frac{P_{S0}^{VL}}{P_T^{VL}}$ is calculated for $(S_0, T) = (63, 1)$. For this composition, $V_{HL} = -8.3807\mu V$, $P_{VL} = 0.558nW$, and consequently $\frac{P_{S0}^{VL}}{P_T^{VL}} = 1.229 \times 10^{-5} \gg 1$. The consequence of (10) for any combination of S_0 and T is $P_{VL} \approx P_{VL} - S_0 \times P_{S0}^{VL}$. Note that for $S_1 = 0$, $P_T^{SL} \neq P_T^{VL}$ and the increase of S_0 increases $\frac{P_{SL}}{P_{VL}}$ as shown in Table 6.

Table 7 compares $\frac{P_{SL}}{P_{VL}}$ for various compositions of S_0 and T. In a $1 \times n$ crosspoint array, for $S_1 = 0$, the power consumption during SL operation is the same for both combinations $(S_0, T) = (n-k, k)$ and $(S_0, T) = (k, n-k)$ where $1 \leq k < n$. In other words, if the voltage on the common nanowire of the crosspoint array for $(S_0, T) = (n-k, k)$ is $\hat{V}_{HL}$, this voltage for $(S_0, T) = (k, n-k)$ would become $-\hat{V}_{HL}$. For example, for $(S_0, T) = (2, 6)$ $\hat{V}_{HL} = -51.096mV$, but for $(S_0, T) = (6, 2)$ $\hat{V}_{HL} = 51.096mV$. In both compositions, $P_{SL} = 4.246nW$, as shown in Table 7. This circuit behavior causes $\frac{P_{SL}}{P_{VL}}$ to increase as S_0 approaches n. As S_0 increases, the voltage drop across the target memristors during SL operation becomes larger than the voltage drop across the target memristors during VL operation. Since all target memristors show the same resistance when reverse biased, $\frac{P_{SL}}{P_{VL}} > 1$. Recall that in SL operation, all source memristors are connected to V^+, whereas in VL operation, under the assumption of $S_1 = 0$, all source memristors are only set to 0 V. Moreover, the power consumption in source memristors during VL operation is negligible as shown in (10); therefore, the increase of S_0 slightly affects P_{VL}. However, the power consumption in source memristors and in load resistor R_G during SL operation can significantly increase the $\frac{P_{SL}}{P_{VL}}$ ratio. When $S_0 = T$, during SL operation $\left|V^+ - \hat{V}_{HL}\right| = \left|V^- - \hat{V}_{HL}\right|$, and thus $\hat{V}_{HL} = 0V$ and $P_{RG} = 0V$. In other words, $P_T^{SL} = P_{S0}^{SL}$. In this case, $\frac{P_{SL}}{P_{VL}} \approx 2$. For example, in a 1×8 crosspoint array for $(S_0, T) = (4, 4)$ $\frac{P_{SL}}{P_{VL}} = 2.113$, or in a 1×64 crosspoint array for $(S_0, T) = (32, 32)$ $\frac{P_{SL}}{P_{VL}} = 2.005$.

In a 1×8 crosspoint array, for any composition of S_0 and T, $\frac{P_{SL}}{P_{VL}} > 1$, i.e., the lower bound of $\frac{P_{SL}}{P_{VL}}$ is 1.025, computed for $(S_0, T) = (1, 7)$. This composition maximizes P_{VL}, while minimizing P_{SL} as shown in Table 7. However, in a $1 \times n$ crosspoint array where $n > 8$, the decrease of k in $(S_0, T) = (k, n-k)$ can bring $\frac{P_{SL}}{P_{VL}}$ to less than 1, as shown in Table 8. Table 8 shows various compositions of S_0 and T for which $\frac{P_{SL}}{P_{VL}} < 1$. In each composition, for a given S_0, only the lower bound of T for which $\frac{P_{SL}}{P_{VL}} < 1$ is presented, i.e., a larger T leads to a smaller $\frac{P_{SL}}{P_{VL}}$.

5.3 Analysis of Power Consumption and Switching Delays in a 1×64 Crosspoint Arrays for $S_1 > 0$

Tables 9, 10, 11, and 12 compare the switching delay and the overall power consumption in a 1×64 crosspoint array during SL and VL for various compositions of S_1, S_0, and T, where $S_1 > 0$. The simulation results, shown in Tables 9, 10, 11, and

Table 7 Comparison of power ratio $\frac{P_{SL}}{P_{VL}}$ of a multi-input multi-output NOR realized in a 1×8 cross point array with $S_1 = 0$ and various values of S_0 and T

(S_1, S_0, T)	Op	P_{S0} (nW)	P_T (nW)	P_{RG} (nW)	P_{avg} (nW)	$\frac{P_{SL}}{P_{VL}}$
(0, 1, 7)	VL	0.027	0.550	0	3.877	1.025
	SL	0.731	0.407	0.392	3.972	
(0, 2, 6)	VL	0.005	0.554	0	3.352	1.267
	SL	0.671	0.455	0.174	4.246	
(0, 3, 5)	VL	0.002	0.556	0	2.786	1.582
	SL	0.613	0.505	0.044	4.408	
(0, 4, 4)	VL	0.556e-3	0.556	0	2.226	2.113
	SL	0.588	0.558	0	4.704	
(0, 5, 3)	VL	0.200e-3	0.557	0	1.672	2.636
	SL	0.505	0.613	0.044	4.408	
(0, 6, 2)	VL	0.062e-3	0.557	0	1.114	3.811
	SL	0.455	0.671	0.174	4.246	
(0, 7, 1)	VL	0.011e-3	0.557	0	0.557	6.427
	SL	0.407	0.731	0.392	3.972	

The parameters are defined in the caption of Table 5

Table 8 A multi-input multi-output NOR with power ratio $\frac{P_{sl}}{P_{vl}} < 1$ for some combinations of S_0 and T with $S_1 = 0$ realized in a $1 \times n$ cross point array

(S_1, S_0, T)	Op	P_{S0} (nW)	P_T (nW)	P_{RG} (nW)	P_{avg} (nW)	$\frac{P_{SL}}{P_{VL}}$
(0, 1, 8)	VL	0.035	0.549	0	4.427	0.987
	SL	0.757	0.388	0.508	4.369	
(0, 2, 12)	VL	0.020	0.551	0	6.652	0.997
	SL	0.818	0.347	0.830	6.63	
(0, 3, 16)	VL	0.016	0.552	0	8.88	0.990
	SL	0.869	0.315	1.147	8.794	
(0, 4, 20)	VL	0.014	0.552	0	11.096	0.982
	SL	0.912	0.290	1.447	10.895	
(0, 5, 23)	VL	0.012	0.552	0	12.756	0.992
	SL	0.933	0.278	1.601	12.66	
(0, 6, 26)	VL	0.010	0.553	0	14.438	0.998
	SL	0.951	0.268	1.742	14.416	
(0, 7, 30)	VL	0.010	0.553	0	16.66	0.987
	SL	0.982	0.253	1.987	16.451	

All of the parameters are defined in the caption of Table 5

12, are consistent with the first four observations stated for a 1×8 crosspoint array in Section 5.1. However, unlike the last two observations stated for a 1×8 crosspoint array, the following is observed.

(1) The main contributors to the power ratio $\frac{P_{SL}}{P_{VL}}$ are S_0, T, and P_{RG}. For example, holding T constant while increasing S_0, as shown in Table 9, decreases the power ratio $\frac{P_{SL}}{P_{VL}}$. In contrast, holding S_0 constant while increasing T, as shown in Table 10, brings $\frac{P_{SL}}{P_{VL}}$ close to 1.

(2) The switching delay t_d in VL is shorter than in SL if $\frac{P_{SI}^{SL}}{P_{S1}^{VL}} > 1$ and is equal or longer than in SL if $\frac{P_{SI}^{SL}}{P_{S1}^{VL}} \leq 1$.

In a 1×64 crosspoint array, for any composition of S_1, S_0, and T where $S_1 > 0$ Eqs. (5), (7), and (8) hold and since $7.07\ 8 \leq \frac{P_{RG}}{P_T^{SL}} \leq 8.328$ Eq. (6) can still approximate the relation between P_{RG} and P_T^{SL} during SL operation. As a result, $\frac{P_{SL}}{P_{VL}}$ can still be approximated by (9). All derivations are in the same manner as shown in Section 5.1.

Table 9 shows that VL consumes more power than SL when $S_0 > 33$. In this case, an increase of T brings the $\frac{P_{SL}}{P_{VL}}$ ratio close to 1 as predicted by (9). This prediction is varified in Table 10. The converse is also true, when $S_0 < 33$, VL consumes less power than SL and the increase of T brings the $\frac{P_{SL}}{P_{VL}}$ ratio close to 1, as shown in Table 11. Note that the increase of S_1 slightly affects the $\frac{P_{SL}}{P_{VL}}$ ratio as shown in Table 12.

In terms of switching delay, volistor gates perform faster than stateful gates when $\frac{P_{SI}^{SL}}{P_{S1}^{VL}} > 1$. When $\frac{P_{SI}^{SL}}{P_{S1}^{VL}} \approx 1$, the propagation delay in both operations is almost the same; when $\frac{P_{SI}^{SL}}{P_{S1}^{VL}} < 1$, stateful gates perform faster than volistors. In other words, the switching delay changes with the voltage on the

Table 9 Comparison of delay t_d and power ratio $\frac{P_{sl}}{P_{vl}}$ of a multi-input NOR realized in a 1×64 crosspoint array with $S_0 \geq 33$ and varied S_1

(S_1, S_0, T)	Op.	P_{S0} (nW)	P_{S0} (nW)	P_T (nW)	P_{RG} (nW)	t_d (ns)	P_{avg} (nW)	$\frac{P_{SL}}{P_{VL}}$
(30, 33, 1)	VL	0.757e-3	0.556	2.228	0	4.041	20.599	1.009
	SL	0.770e-3	0.770e-6	2.228	18.541	4.041	20.792	
(29, 34, 1)	VL	0.857e-3	0.556	2.227	0	4.034	21.156	0.982
	SL	0.824e-3	0.824e-6	2.227	18.539	4.042	20.767	
(25, 38, 1)	VL	1.423e-3	0.558	2.227	0	4.048	23.467	0.885
	SL	1.107e-3	1.107e-6	2.227	18.533	4.044	20.760	
(13, 44, 1)	VL	0.007	0.554	2.222	0	4.072	26.689	0.778
	SL	0.004	4.069e-6	2.224	18.484	4.060	20.760	
(1, 62, 1)	VL	2.021	0.492	2.098	0	4.907	34.623	0.583
	SL	0.579	0.579e-3	2.159	17.406	4.470	20.18	

The parameters are defined in the caption of Table 5

Table 10 Comparison of delay t_d and power ratio $\frac{P_{sl}}{P_{vl}}$ of a multi-input NOR realized in a 1 × 64 crosspoint array with constant input values S_0 and S_1, and varied

(S_1, S_0, T)	Op.	P_{S1} (nW)	P_{S0} (nW)	P_T (nW)	P_{RG} (nW)	t_d (ns)	P_{avg} (nW)	$\frac{P_{SL}}{P_{VL}}$
(1, 40, 1)	VL	0.908	0.513	2.141	0	4.575	23.569	0.856
	SL	0.603	0.603e-3	2.157	17.383	4.461	20.167	
(1, 40, 4)	VL	1.179	0.507	2.129	0	4.665	29.975	0.888
	SL	0.821	0.821e-3	2.145	17.187	4.543	26.621	
(1, 40, 8)	VL	1.592	0.5	2.113	0	4.780	38.496	0.914
	SL	1.161	1.161e-3	2.130	16.928	4.65	35.175	
(1, 40, 16)	VL	2.592	0.484	2.081	0	5.049	55.248	0.943
	SL	2.005	2.005e-3	2.098	16.421	4.895	52.074	

The parameters are defined in the caption of Table 5

common nanowire of a crosspoint array or equivalently the voltage drop across the target memristors. A logic gate operates faster if the voltage drop across its target memristors is larger. The increase of S_0 versus the decrease of S_1 results in faster stateful gates and vice versa as shown in Table 9. In addition, the increase of T versus the decrease of S_1 results in faster volistor gates. Table 11 shows the increase of t_d in SL versus VL for constant S_0 with T varied from 1 to 57.

Table 9 shows $\frac{P_{SL}}{P_{VL}} < 1$ when $S_0 > 33$, regardless of value S_1. Table 12 reinforces our conclusion that S_1 has almost no effect on the power ratio, $\frac{P_{SL}}{P_{VL}}$. The increase of S_0 and the decrease of S_1 result in $P_{S1}^{VL} > P_{S1}^{SL}$ and thus a longer switching delay t_d for VL.

Table 10 shows how the increase of T affects $\frac{P_{SL}}{P_{VL}}$ when $S_0 > 33$. The increase of T brings $\frac{P_{SL}}{P_{VL}}$ close to 1. For all compositions, t_d is longer for VL. This long switching delay is related to the increase of T and the large S_0.

Table 11 shows how the increase of T affects $\frac{P_{SL}}{P_{VL}}$ when S_1 and S_0 are fixed at at 1 and 6, respectively. For all compositions, shown in Table 11, $\frac{P_{RG}}{S_0 \times P_{S0}^{VL}} \approx 5.2$; however, the increase of T brings $\frac{P_{SL}}{P_{VL}}$ close to 1, reducing the effect of $\frac{P_{RG}}{S_0 \times P_{S0}^{VL}}$ in $\frac{P_{SL}}{P_{VL}}$ significantly as predicted by (9). As a result, executing a multi-input multi-output gate in a small crosspoint array, e.g., 1 × 8, with VL is more power efficient than with SL. This conclusion is consistent with (9) and with the simulation results shown in Table 5. The increase of T results in $P_{S1}^{SL} > P_{S1}^{VL}$ and thus longer t_d for SL.

Table 12 shows how the increase of S_1 affects $\frac{P_{SL}}{P_{VL}}$ when S_0 (> 33) and T are fixed at 40 and 1, respectively. The increase of S_1 has almost no effect on $\frac{P_{SL}}{P_{VL}}$, i.e., all compositions result in almost the same power ratio of 0.856. Although the increase of S_1 has almost no effect on $\frac{P_{SL}}{P_{VL}}$, it does decrease the switching delay in both SL and VL operations.

As a summary, in a 1 × 8 crosspoint array, the power consumption in SL is higher than in VL and volistors operate faster than stateful gates. In a 1 × 64 crosspoint array, when $S_1 > 0$, the majority of the power consumption in VL is $S_0 \times P_{S0}^{VL} + T \times P_T$, and in SL is $P_{RG} + T \times P_T$ as described in (5), (6), (7), and (8). In this case, $\frac{P_{SL}}{P_{VL}}$ can be approximated with (9). In a 1 × 64 crosspoint array, a multi-input volistor gate consumes less power than a multi-input stateful gate, unless the majority

Table 11 Comparison of delay t_d and power ratio $\frac{P_{sl}}{P_{vl}}$ of a multi-input NOR realized in a 1 × 64 crosspoint array with constant number of inputs and varied outputs

(S_1, S_0, T)	Op.	P_{S1} (nW)	P_{S0} (nW)	P_T (nW)	P_{RG} (nW)	t_d (ns)	P_{avg} (nW)	$\frac{P_{SL}}{P_{VL}}$
(1, 6, 1)	VL	0.035	0.549	2.212	0	4.103	5.541	3.636
	SL	0.643	0.643e-3	2.155	17.345	4.454	20.147	
(1, 6, 5)	VL	0.140	0.54	2.195	0	4.223	14.355	2.002
	SL	0.960	0.96e-3	2.139	17.074	4.571	28.735	
(1, 6, 15)	VL	0.693	0.519	2.152	0	4.474	36.087	1.383
	SL	2.020	2.020e-3	2.098	16.420	4. 874	49.922	
(1, 6, 30)	VL	2.263	0.489	2.090	0	4.942	67.897	1.192
	SL	4.247	4.247e-3	2.040	15.482	5.411	80.954	
(1, 6, 57)	VL	7.104	0.439	1.985	0	6.026	122.883	1.096
	SL	10.069	10.069e-3	1.94	13.925	6.681	134.634	

The parameters are defined in the caption of Table 5

Table 12 Comparison of delays t_d and power ratio $\frac{P_{sl}}{P_{vl}}$ of a multi-input NOR realized in a 1 × 64 crosspoint array with varying S_1 and $S_0 > 33$

(S_1, S_0, T)	Op.	P_{S1} (nW)	P_{S0} (nW)	P_T (nW)	P_{RG} (nW)	t_d (ns)	P_{avg} (nW)	$\frac{P_{SL}}{P_{VL}}$
(1, 40, 1)	VL	0.908	0.513	2.141	0	4.585	23.569	0.856
	SL	0.603	0.603e-3	2.157	17.383	4.468	20.167	
(5, 40, 1)	VL	0.039	0.548	2.212	0	4.140	24.327	0.850
	SL	0.027	0.027e-3	2.215	18.326	4.122	20.677	
(10, 40, 1)	VL	0.010	0.553	2.221	0	4.079	24.441	0.849
	SL	0.007	0.007e-3	2.223	18.456	4.080	20.749	
(20, 40, 1)	VL	0.002	0.555	2.226	0	4.064	24.466	0.849
	SL	1.727e-3	1.727e-6	2.226	18.52	4.057	20.781	

The parameters are defined in the caption of Table 5

of inputs are logic "0." When $S_1 = 0$, a multi-input volistor gate consumes less power than a multi-input stateful gate. In a 1 × 64 crosspoint array, for $S_1 > 0$, a logic gate with a large fan-out consumes the same power when realized with SL and VL. When $S_1 = 0$, the majority of the power consumption in VL is $T \times P_T^{VL}$ as shown in (10). However, in SL operation, the power consumption in each circuit element depends on the combination of S_0 and T.

6 Memory Application

Memristors are nanoscale nonvolatile devices used primarily as resistive memory. The generic structure for memristive memory is a crossbar array. The crossbar arrays can be vertically stacked creating a 3D crossbar memory [16]. The potential problem of the crossbar structure is the leakage pathways from surrounding memory cells. In a recent work, Vourkas et al. [17] proposed memories comprised of parallel/serial complementary memristive switches. In order to have states that are distinguishable, they introduced insulating junctions within a crossbar array to restrain leakage pathways [18].

In another approach, Kim et al. [4] fabricated a crossbar array utilizing rectifying memristors rather than standard memristors. The use of rectifying memristors eliminates the leakage pathways due to the diode-like behavior of memristors, as discussed in Section 2. Furthermore, the large $\frac{R_{OPEN}}{R_{CLOSED}}$ ratio of the rectifying memristors allows for multi-level memory as demonstrated in [4]. This memory array can be programmed exploiting volistor gates. As demonstrated in Section 5, volistor gates are more efficient than stateful gates in terms of power consumption. In the process of programming the memory array, first, all memristors are initialized to low resistance state. This initialization step can be implemented in one clock cycle. Second, every cell of the memory array is programmed sequentially. The latter approach for memory realization is more efficient than the former approach in terms of simplicity, power consumption, reliability, and robustness, e.g., the second approach allows for multi-level memory realization. Furthermore, realization of memory array with rectifying memristors allows for programming the memory cells to exploit volistors, which is a more power-efficient approach than the conventional stateful approach. This distinction between memories is originated from the property of rectifying memristors over standard memristors.

Volistors can be used to create memory-like digital sensor integrating devices. These devices read voltages from many sensors in parallel/serial mode and convert them to words of stored resistive information. The stored information can be next processed by combinational stateful memristive circuits.

7 Conclusion

Volistors capitalize on the characteristics of rectifying memristors to allow voltages to be used directly as inputs. There is no need to store inputs as resistances in memristors before logic can be performed. This eliminates the use of R_G to read out the inputs in the first logic level and thus eliminates the most power-hungry circuit element in stateful logic. There is no static power in our circuits. The switching power consumed by volistor logic in a crosspoint array is entirely due to the leakage through the reverse biased rectifying memristor as there is no direct path to the ground. Using volistors at the first level provides NOR and AND directly which provides a flexibility sufficient to remove the need for IMP gates in stateful logic and to realize all subsequent levels with only NOR/NOT. Realizing all stateful logic levels with NOR/NOT simplifies the driver circuitry and removes the need for keeper circuits. An additional advantage is that the first-level volistor logic can have as many inputs as the number of available memristors in the crosspoint array within a single operation. However, inputs set to 0V incur a higher power penalty than the inputs set to V^+ or V^-. The two-dimensional, multi-output property of volistors provides for both a large fan-out (at the cost of increased power consumption) and for a great flexibility

in the placement of calculation results. Using multiple crossbars in a network provides parallelization which leads to pipelined and SIMD architectures in the data path. Several different first-level volistor operations can be computed in separate crossbars at the same time, greatly speeding up what is typically the most complex level of Boolean function implementations. Subsequent logic levels can be accomplished in connected crossbars in an approach that uses mixed-input gates and stateful NOR. This hybrid approach with volistors and stateful NOR can be used to compute logic faster, with less power and with simpler external control circuitry than a solely stateful approach.

Acknowledgments This work was supported by the Higher Committee for Education Development in Iraq (HCED).

References

1. Borghetti, J., Snider, G., Kuekes, P., Yang, J., Stewart, D., Williams, R. (2010). Memristive' switches enable 'stateful' logic operations via material implication. *Nature, 464*(7290), 873–876.
2. Lehtonen, E., Poikonen, J. H., Laiho. M. (2012). Applications and limitations of memristive implication logic. CNNA. doi:10.1109/CNNA.2012.6331438.
3. Kim, K., Shin, S., Kang, S. (2011). Field programmable stateful logic array. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 30*(12), 1800–1813.
4. Kim, K., Gaba, S., Wheeler, D., Cruz-Albrecht, J., Hussain, T., Srinivasa, N., Lu, W. (2012). A functional hybrid memristor crossbar-array/CMOS system for data storage and neuromorphic applications. *Nano Letters, 12*(1), 389–395.
5. Kvatinsky, S., Wald, N., Satat, G., Kolodny, A., Weiser, U.C., Friedman, E.G. (2012). MRL—memristor ratioed logic. CNNA. doi:10.1109/CNNA.2012.6331426.
6. Kim, K. H., Hyun Jo, S., Gaba, S., Lu, W. (2010). Nanoscale resistive memory with intrinsic diode characteristics and long endurance. *Applied Physics Letters, 96*(5), 053106.
7. Lehtonen, E., Tissari, J., Poikonen, J., Laiho, M., Koskinen, L. (2014). A cellular computing architecture for parallel memristive stateful logic. *Microelectronics Journal, 45*(11), 1438–1449.
8. Lee, M. J., Lee, C. B., Lee, D., Lee, S. R., Chang, M., Hur, J. H., Kim Y B, Kim C J, Seo D H, Seo S, Chung UI. (2011). A fast, high-endurance and scalable non-volatile memory device made from asymmetric Ta2O5− x/TaO2− x bilayer structures. *Nature Materials, 10*(8), 625–630.
9. Torrezan, A. C., Strachan, J. P., Medeiros-Ribeiro, G., Williams, R. S. (2011). Sub-nanosecond switching of a tantalum oxide memristor. *Nanotechnology, 22*(48), 485203.
10. Lehtonen, E., Poikonen, J., Laiho, M. (2010). Two memristors suffice to compute all Boolean functions. *Electronics Letters, 46*(3), 230.
11. Raghuvanshi, A., Perkowski, M., (2014). Logic synthesis and a generalized notation for memristor realized material implication gates. ICCAD. doi:10.1109/ICCAD.2014.7001393.
12. Gimpel, J. F. (1967). The minimization of TANT networks. *IEEE Transactions on Electronic Computers, 16*(1), 18–38.
13. Laiho, M. Lehtonen, E. (2010). Cellular nanoscale network cell with memristors for local implication. ISCAS. doi:10.1109/ISCAS.2010.5537188.
14. Perkowski, M., Fiszerl, R., Kerntopf, P., Lukac, M. (2012). An approach to synthesis of reversible circuits for partially specified functions. NANO. doi:10.1109/NANO.2012.6322122.
15. Jo, S., Kim, K., Lu, W. (2009). High-density crossbar arrays based on a SI memristive system. *Nano Letters, 9*(2), 870–874.
16. Strukov, D. B., & Williams, R. S. (2009). Four-dimensional address topology for circuits with stacked multilayer crossbar arrays. *Proceedings of the National Academy of Sciences, 106*, 20155–20158.
17. Vourkas, I., & Sirakoulis, G. C. (2014). Nano-crossbar memories comprising parallel/serial complementary memristive switches. *BioNano Science, 4*, 166–179.
18. Vourkas, I., Stathis, D., Sirakoulis, G. C. (2013). Improved read voltage margins with alternative topologies for memristor-based crossbar memories. VLSI. doi:10.1109/VLSI-SoC.2013.6673304.