Memristor-based IMPLY Logic Design Procedure

Shahar Kvatinsky, Avinoam Kolodny,
and Uri C. Weiser
Department of Electrical Engineering
Technion – Israel Institute of Technology
Haifa 32000 ISRAEL
{skva@tx, kolodny@ee, uri.weiser@ee}.technion.ac.il

Eby G. Friedman
Department of Electrical and Computer Engineering
University of Rochester
Rochester, NY 14627 USA
friedman@ece.rochester.edu

Abstract — **Memristors can be used as logic gates. No design methodology exists, however, for memristor-based combinatorial logic. In this paper, the design and behavior of a memristive-based logic gate – an IMPLY gate - are presented and design issues such as the tradeoff between speed (fast write times) and correct logic behavior are described, as part of an overall design methodology. A memristor model is described for determining the write time and state drift. It is shown that the widely used memristor model - a linear ion drift memristor - is impractical for characterizing an IMPLY logic gate, and a different memristor model is necessary such as a memristor with a current threshold.**

Keywords - memristor; memristive systems; IMPLY; design methodology; logic

I. INTRODUCTION

Memristors are passive elements with varying resistance (also known as a memristance), conceived theoretically in [1]. Changes in the memristance depend upon the history of the device, the total charge which passes through it, or, alternatively, the total flux in the device (the integral over time of the applied voltage at the ports of the device).

In 2008, Hewlett-Packard announced the fabrication of a working memristor [2]. A linear ion drift model was proposed for describing the behavior of this memristor. The memristance of a linear ion drift memristor is

$$M(q) = R_{OFF}\left(1 - \frac{\mu_v R_{ON}}{D^2} q(t)\right), \tag{1}$$

where R_{OFF} and R_{ON} are, respectively, the maximum and minimum resistance of the memristor, μ_v is the average ion mobility, D is the memristor physical thickness, and $q(t)$ is the total charge passing through the memristor. The linear ion drift model is the most commonly used memristor model, although practical memristors exhibit highly non-linear behavior.

Memristors can be used for numerous applications, such as memory [3], neuromorphic systems [4], and analog circuits (e.g., see [5]). One interesting application of memristors is logic, using memristors as building blocks of logic gates. To use memristors in a digital manner, a high memristance is considered as logic *0* and a low memristance is considered as logic *1*. Several approaches for memristor-based logic have been proposed, e.g., [6] and [7], which suggest using memristors as configurable switches as in an FPGA. The logic gates are designed as CMOS gates or as programmable majority logic array (PMLA) based on Goto pairs as logic gates [8].

Another approach is to use memristors as the primary building blocks of a logic gate. Each memristor acts as an input, output, computational logic element, and a latch in different stages of the computing process [9]. In [10], a memristor-based logic gate - the IMPLY gate, is presented. Since this logic function together with FALSE (a function that always yields the value *0* as an output) comprise a computationally complete logic structure, it may potentially provide a basic logic element for a memristor-based circuit. The truth table for *p IMPLY q* is listed in Table 1. Unlike CMOS logic [11], no design methodology exists for memristor-based logic circuits.

In this paper, a design methodology is suggested for memristor-based IMPLY logic gates. A memristor-based IMPLY gate and related limitations are also presented here. The tradeoff between performance and robustness is described as well as the necessity to refresh the logic gate.

This paper is organized as follows. In Section II, the operation of a memristor-based IMPLY gate is described. In section III, the performance and limitations of this logic gate are presented. In section IV, a design example is described, and simulation results of the IMPLY gate are shown. The paper is summarized in section V.

II. MEMRISTOR-BASED IMPLY GATE

The logic function $p \rightarrow q$ (also known as "*p IMPLIES q*," "*material implication*," and "*if p then q*") is described in [10]. The proposed memristor logic is based upon a resistor R_G ($R_{ON} < R_G < R_{OFF}$) connected to two memristors, named *P* and *Q*, acting as digital switches. The corresponding initial memristances *p* and *q* are the inputs of the gate; while the output of the gate is the final memristance of *Q* (the result is written into the logic state *q*). A schematic of an IMPLY gate is shown in Figure 1.

The basic concept is to apply different negative voltages to *P* and *Q*, where V_{SET}, the applied voltage on *Q*, has a higher magnitude than V_{COND}, the applied magnitude on *P* ($|V_{COND}| < |V_{SET}|$). If $p = 1$ (low resistance), the voltage on the common terminal is approximately V_{COND} and the voltage on

This work was partially supported by Hasso Plattner Institute and by Intel grant "Heterogeneous Computing, the Inevitable Solution: Power Management, Scheduling and ISA" grant no. 864-737-13.

the memristor Q is approximately V_{SET} - V_{COND}, which is sufficiently small to maintain the logic state of q. In the case of $p = 0$ and $q = 0$ (high resistances), the applied voltage on Q is approximately V_{SET} and Q is switched *ON* ($q = 1$). In the case of $p = 0$ and $q = 1$, the logic state of q is maintained.

A two input NAND, based on a memristor-based IMPLY gate and a FALSE logic gate, is described in [10]. The circuit is comprised of three memristors; the operation of this NAND gate changes the function of each memristor during the computing process. Two memristors act as inputs in the initial stage, one memristor acts as the output in the last stage, and all memristors act together as a computational logic element (as a memristor-based IMPLY gate) during different stages of the computing process. This application requires three computing stages (one FALSE and two IMPLY). A schematic and the sequence of an IMPLY-based NAND are shown in Figure 2.

The execution of any general Boolean function f: $B^n \rightarrow B$ can be implemented with only $n + 3$ memristors [12], where three additional memristors carry out the computation. Only two memristors are required for up to three inputs. Computation of the function is performed in steps. In each step, either FALSE is applied to one memristor, or an IMPLY is applied to two memristors, where the output is written (which is one of the inputs of the computational IMPLY stage). This process requires a long sequence of operations depending upon the number of inputs. This methodology is improved in [13] where only two additional memristors are used rather than three. While [12] and [13] present a general algorithm to compute any Boolean function with a minimal number of memristors, the computational process requires a large number of functional stages, and therefore requires significant computational time.

III. Design Considerations and performance Analysis of the Memristor-Based IMPLY Gate

A. *Analysis fundamentals*

The behavior of a memristor-based IMPLY gate is mathematically cumbersome for analysis. There is therefore a need to develop heuristics for designing memristive circuits.

These heuristics can be extended to enable a complete design methodology for memristor-based circuits. A flow diagram of an IMPLY logic gate design methodology is shown in Figure 3.

In this section, design strategies for choosing the proper circuit parameters (R_G, V_{SET}, and V_{COND}) are discussed. The tradeoff between the delay time of the circuit (to maintain the proper write time) and the number of cycles to refresh the memristors (because of state variable drift) is described.

TABLE 1. Truth table of IMPLY function.

Case	*p*	*q*	*p→q*
1	0	0	1
2	0	1	1
3	1	0	0
4	1	1	1

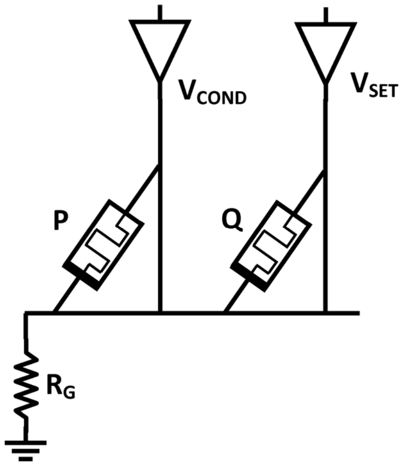
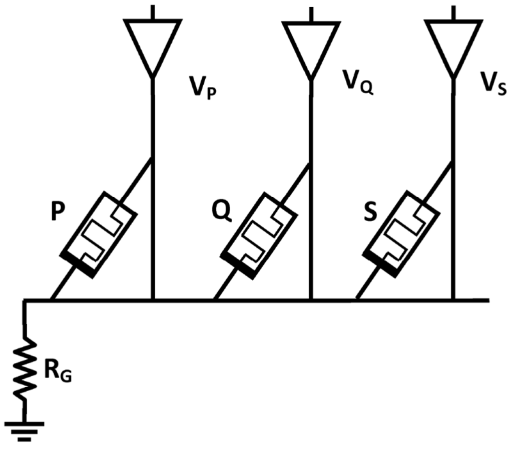


Figure 1. Schematic of a memristor-based IMPLY gate. Two memristors *P* and *Q* are connected to a resistor R_G. The logic state of the memristors *P* and *Q* are, respectively, *p* and *q*.

Step	Voltages		
Step 1: *s=0*			$V_S = V_{CLEAR}$
Step 2: *p→s*	$V_P = V_{COND}$		$V_S = V_{SET}$
Step 3: *q→s*	$V_q = V_{COND}$	$V_S = V_{SET}$	

(a)



(b)

Figure 2. IMPLY NAND logic gate. (a) Logical operation of an IMPLY-based NAND, the logic gate requires three sequential steps, and (b) schematic of IMPLY-based NAND gate.

B. *The tradeoff between performance and robustness*

V_{SET} and V_{COND}, the applied voltages on P and Q, are fixed. Therefore, for any initial state, the memristor state q tends to drift towards the *ON* state. For digital operation, the state of q should either stay unchanged or switch fully *ON* (changing the logic state from logic *0* to logic *1*).

The different input combinations are presented in Table 1. Note that in cases 2 and 4, the initial state of q is logic *1* and the logic gate output q is also logic *1*. The gate operation, therefore, electrically reinforces the logic state of q, and the memristance of Q is reduced.

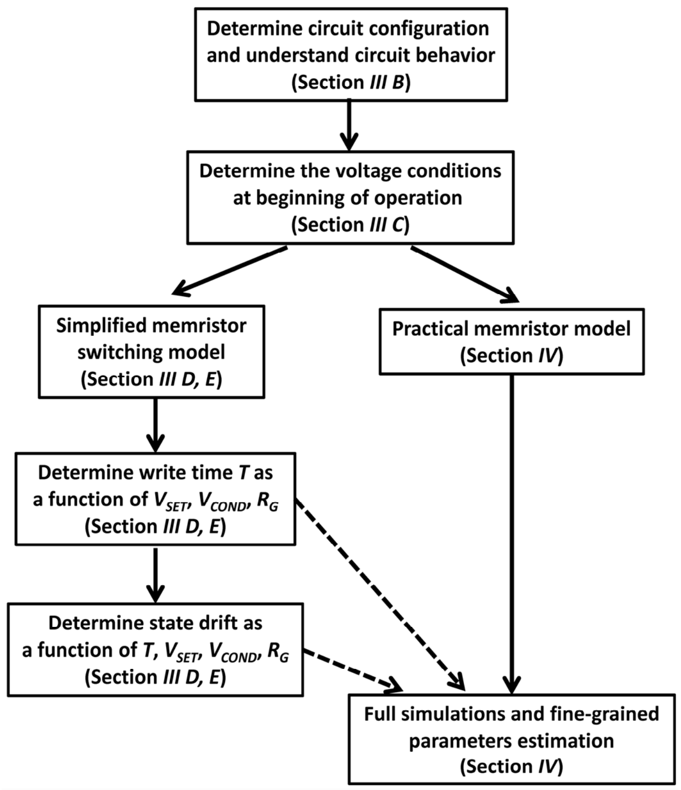

Determine circuit configuration and understand circuit behavior (Section *III B*)
Determine the voltage conditions at beginning of operation (Section *III C*)
Simplified memristor switching model (Section *III D, E*)
Practical memristor model (Section *IV*)
Determine write time *T* as a function of V_{SET}, V_{COND}, R_G (Section *III D, E*)
Determine state drift as a function of *T*, V_{SET}, V_{COND}, R_G (Section *III D, E*)
Full simulations and fine-grained parameters estimation (Section *IV*)

Figure 3. IMPLY logic gate design flow diagram. Each box refers to the relevant section of this paper.

In case 1, the initial state of q is logic *0*; after applying the external voltages, q is switched *ON*. This case determines the time required to apply V_{SET} and V_{COND} until the logic state of q reaches the desired state (above a certain level of conduction to maintain correct logic behavior). This case determines the speed of the circuit in terms of the write time.

In case 3, the initial state of q is logic *0*. This logic state should remain unchanged after applying V_{SET} and V_{COND}, although the voltages tend to change the internal state of q towards the *ON* state of logic *1*. This phenomenon is "state drift." The logic *0* state of q, which is the output of the gate, is electrically "weaker" than the input logic state of q (the memristance of q after applying the voltages is lower than the initial memristance). State drift may require refreshing the state; otherwise, the sensing action may incorrectly switch the logic state of q. State drift depends upon the write time determined for case 1; a long write time increases the state drift phenomenon.

C. Basic principles for parameter determination and design procedure

Although it is difficult to compute the precise value of the applied voltage on Q, it is possible to determine the applied voltage on Q at the beginning of the logic gate activity. The initial applied voltage on Q is different for each case (a different initial memristance for q and p). The initial applied voltages on P and Q are listed in Table 2 under the assumptions that the memristance of logic 1 and logic 0 is, respectively, R_{ON} and R_{OFF}, where $R_{OFF} >> R_{ON}$.

From the initial applied voltages, some necessary conditions for correct logic behavior can be determined. These conditions are not precise, but can provide design constraints. The basic design principle is that the write time of the logic gate is determined from case 1, but the parameters of the circuit should also not exceed a specific state drift in case 3. To determine the circuit parameters, an effective model for the memristors needs to be chosen. The model needs to be sufficiently accurate, while also correctly representing the switching behavior. Inserting the initial applied voltages into the simple memristor switching model can provide an approximate estimate of the circuit parameters.

D. Write time and state drift for a binary memristance

A useful and simple switching model is the binary memristance model. Assume only two allowed memristances, R_{ON} and R_{OFF}. A total charge Q' must flow through the memristor to cause the memristance R_{OFF} to switch to memristance R_{ON}. Under these assumptions and by solving both the switching behavior in case 1 and the write time T as a function of Q', the circuit parameter T is

$$T = \left[\frac{R_{OFF}{}^{2} + 2R_{OFF}R_G}{R_{OFF}V_{SET} + R_G\left[V_{SET} - V_{COND}\right]}\right]\cdot Q'. \qquad (2)$$

The write time for different circuit parameters and a varying V_{SET} is shown in Figure 4. Note that the logic gate is faster with higher applied voltages, or smaller R_{OFF}.

Under this model, it is possible to limit the state drift (case 3) for a fixed drift. The state drift is

$$q_q(T) \approx \left[V_{SET} - \frac{R_G}{R_{ON} + R_G}V_{COND}\right]\cdot\left[\frac{R_{OFF} + 2R_G}{R_{OFF}V_{SET} + R_G\left[V_{SET} - V_{COND}\right]}\right]\cdot Q', \qquad (3)$$

where $q_q(T)$ is the total charge flowing through memristor Q after time T in case 3. To limit the state drift to a value of $Q'/4$, after four times, the logic gate is applied as in case 3, and the state drift changes the memristive logic state. This phenomenon requires a refresh every three times the gate is used, since the logic state changes during the fourth time. The allowed value of V_{SET} for several circuit parameters is shown in Figure 5. Note that the state drift is more significant with a higher applied voltage, or with smaller R_{OFF}. Combining Figures 4 and 5, the tradeoff between the speed and robustness of a memristive logic gate is shown in Figure 6.

E. R_G for a fixed threshold model

Another simple memristor model assumes non-linear behavior with a fixed threshold voltage V_{ON}. For an applied voltage below V_{ON}, the memristance is unchanged. To produce correct logical behavior, the initial applied voltage on Q must be above the threshold voltage in case 1 and below the threshold voltage in case 3. Adding this assumption to the initial applied voltage (see Table 2) leads to the following two conditions on the circuit parameters,

TABLE 2. APPLIED LOGIC GATE VOLTAGES V_Q AND V_P, RESPECTIVELY, ON MEMRISTORS P AND Q AT $t = 0$, UNDER THE ASSUMPTIONS THAT THE MEMRISTANCE OF LOGIC *1* AND LOGIC *0* IS, RESPECTIVELY, R_{ON} AND R_{OFF}, WHERE $R_{OFF} >> R_{ON}$.

Case	$V_Q(t=0)$	$V_P(t=0)$
1	$\frac{R_{OFF}+R_G}{R_{OFF}+2R_G}\cdot V_{SET}-\frac{R_G}{R_{OFF}+2R_G}\cdot V_{COND}$	$-\left[\frac{R_G}{R_{OFF}+2R_G}\cdot V_{SET}-\frac{R_{OFF}+R_G}{R_{OFF}+2R_G}\cdot V_{COND}\right]$
2	$V_{SET}\cdot\frac{R_{ON}}{R_{OFF}}\cdot\frac{R_{OFF}+R_G}{R_{ON}+R_G}\approx V_{SET}$	$-\left[V_{SET}\cdot\frac{R_G}{R_{ON}+R_G}-V_{COND}\right]$
3	$V_{SET}-V_{COND}\cdot\frac{R_G}{R_{ON}+R_G}$	V_{COND}
4	$V_{SET}\cdot\frac{R_{ON}+R_G}{R_{ON}+2R_G}-V_{COND}\cdot\frac{R_G}{R_{ON}+2R_G}$	$-\left[V_{SET}\cdot\frac{R_G}{R_{ON}+2R_G}-V_{COND}\cdot\frac{R_{ON}+R_G}{R_{ON}+2R_G}\right]$

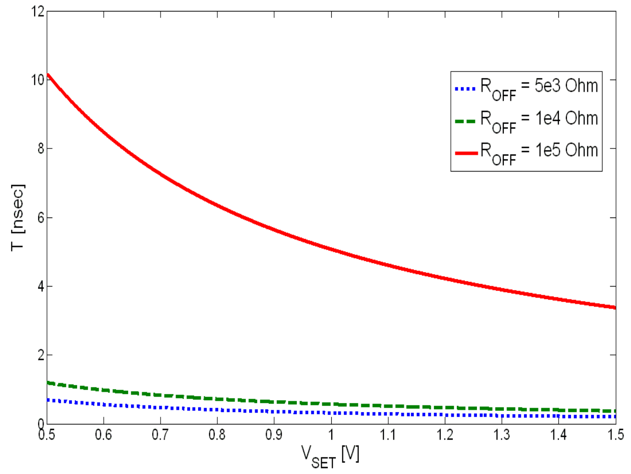


Figure 4. Write time *T* in case 1 for three values of R_{OFF} (5 kΩ, 10 kΩ, and 100 kΩ) under the assumptions of a binary resistance model and Q' = 5·10^{-14} C.

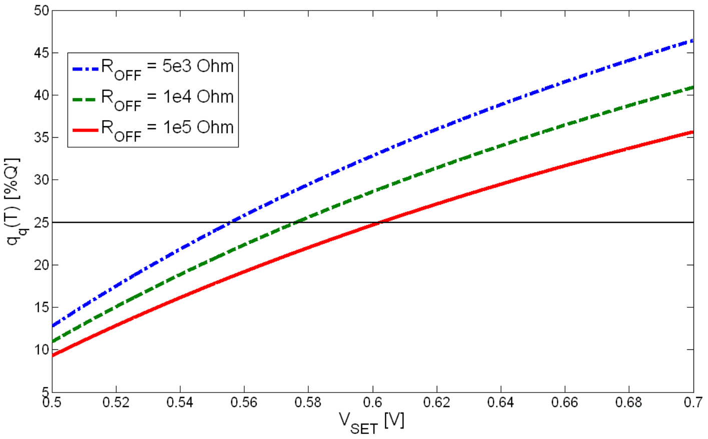


Figure 5. Allowed values of V_{SET} for limited state drift in case 3 of *Q'/4*. V_{SET} is allowed if $q_q(T)$ is smaller than *Q'/4* (the horizontal line in the figure).

$$R_{ON}\cdot\frac{V_{SET}-V_{ON}}{V_{ON}-[V_{SET}-V_{COND}]}<R_G<R_{OFF}\cdot\frac{V_{SET}-V_{ON}}{2V_{ON}-[V_{SET}-V_{COND}]}, \tag{4}$$

$$\frac{V_{SET}}{V_{COND}}<\frac{R_{OFF}}{R_{ON}}. \tag{5}$$

The allowed value for R_G for several circuit parameters and varying V_{SET} are shown in Figure 7.

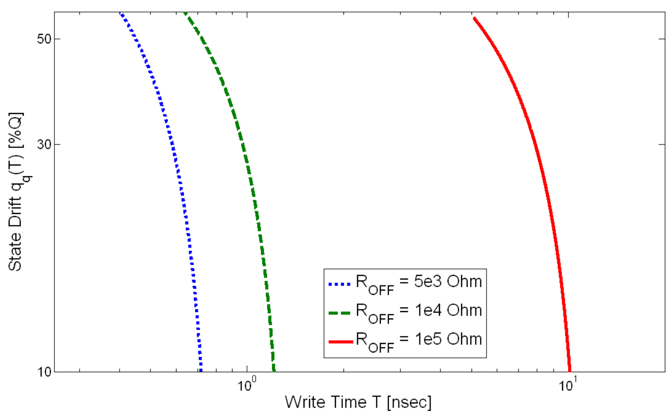


Figure 6. Tradeoff between the logic gate speed (write time) and robustness (the state drift in case 3 for memristor *Q*), for three values of R_{OFF} (5 kΩ, 10 kΩ, and 100 kΩ) under the assumptions of a binary resistance model and Q' = 5·10^{-14} C.

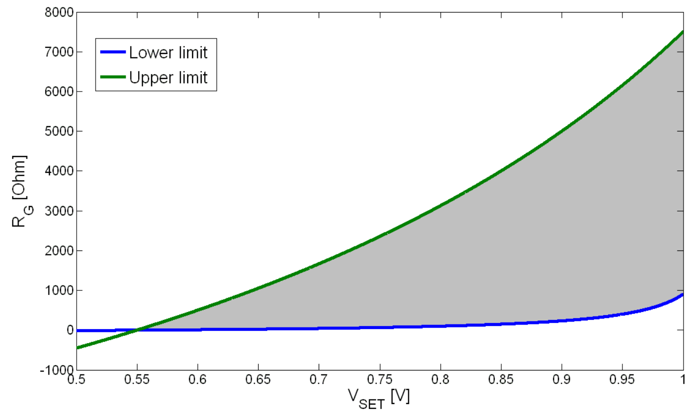


Figure 7. Allowed value of R_G depends on V_{SET}. The upper line is the upper bound for allowed R_G and the lower line is the lower allowed bound for R_G. Under the assumption of a threshold voltage V_{ON} = 0.55 V, V_{COND} = 0.5 V, R_{ON} = 100 Ω, and R_{OFF} = 10 kΩ.

IV. DESIGN EXAMPLE

As a specific example of applying the flow chart of Figure 3, assume the requirements for a circuit are a maximum write time of 0.5 μsec (note that the write time is normalized. A practical memristor write time is significantly faster [14]) and the maximum state drift is $0.025R_{OFF}$ (2.5% of the state drift as compared to full switching).

Assume a memristor with R_{ON} and R_{OFF}, respectively, of 1 kΩ and 100 kΩ. Set one circuit parameter V_{COND} to 0.5 V. The behavior of an ideal IMPLY logic gate (zero write time, no state drift) is shown in Figures 8 and 9. Practical logic gates, however, have non-zero write time and state drift. From Figures 4 and 5, note that as V_{SET} rises, the logic gate write time T decreases and the gate response is faster; however, the state drift phenomenon is more significant. From (5),

$$0.5V < V_{SET} < 50V\,. \tag{6}$$

This expression only produces a lower bound on V_{SET}, since the upper bounds are significantly higher than practical on-chip supply voltages. For a current-controlled memristor, it is unrealistic to determine an exact equivalent voltage threshold (which depends on the transient memristance of the device). A good approximation for an equivalent voltage threshold is

$$V_{ON} = i_{ON} \cdot R_{OFF}, \qquad (7)$$

where V_{ON} is the voltage threshold, and i_{ON} is the current threshold. For a memristor with a current threshold of 7 μA, the equivalent voltage threshold is 0.7 volts. From (4), R_G is

$$1.5k\Omega < R_G < 33.3k\Omega. \qquad (8)$$

The widely used linear ion drift memristor model [15] is incompatible with IMPLY logic gates. In this model, the memristance changes linearly for any applied voltage; the state drift phenomenon is therefore significant, as shown in Figures 10 and 11. Hence, a different memristor model with a current threshold is preferable [16]. With this model, the exact circuit parameters are selected. The chosen circuit parameters are R_{ON} = 1 kΩ, R_{OFF} = 100 kΩ, V_{COND} = 0.5 V, V_{SET} = 1 V, and R_G = 5 kΩ. SPICE simulation results for these parameters are shown in Figures 12 and 13. The write time and state drift for several circuit parameters are listed in Table 3. An increase in the resistance of R_G or decrease in the voltage level of V_{SET} delays the gate, but lowers the state drift (and vice versa).

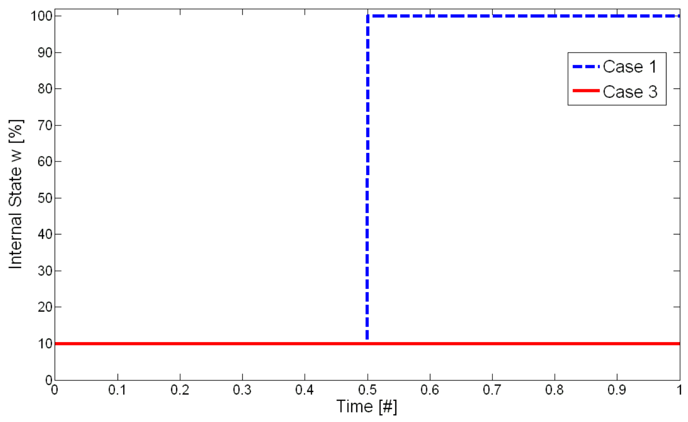


Figure 8. State drift of an ideal IMPLY logic gate. While the logic state in case 1 changes to a zero write time, the drift for case 3 is zero.

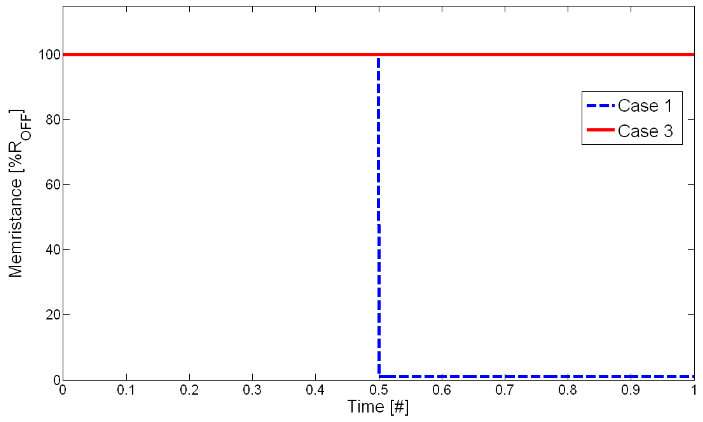


Figure 9. Memristance of an ideal IMPLY logic gate. While the memristance in case 1 decreases to R_{ON} within a zero write time, the memristance in case 3 does not change.

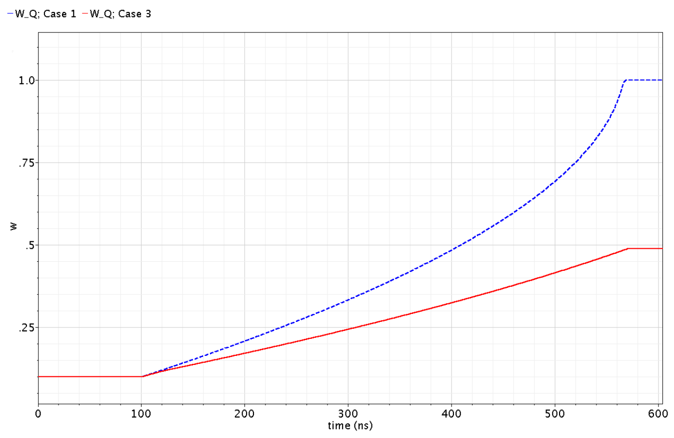


Figure 10. State variable *w* of *q* when applying IMPLY logic gate for cases 1 (dashed line) and 3 (solid line) for a memristor with linear ion drift. *T* is 468.1 nsec. The state drift for case 3 is 48.9%, which makes this model impractical for an IMPLY logic gate.

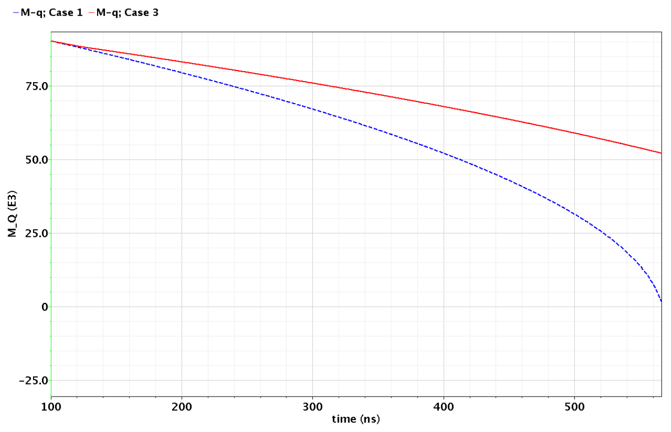


Figure 11. The memristance of *q* when applying an IMPLY logic gate for cases 1 (dashed line) and 3 (solid line) for a memristor with linear ion drift.

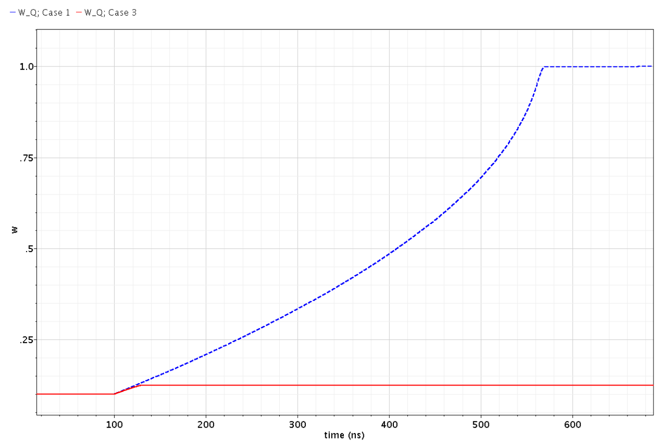


Figure 12. State variable *w* of *q* when applying an IMPLY logic gate for cases 1 (dashed line) and 3 (solid line) for a memristor with a threshold model (current threshold is 7 μA). *T* is 470.3 nsec. The state drift for case 3 is 2.44%.

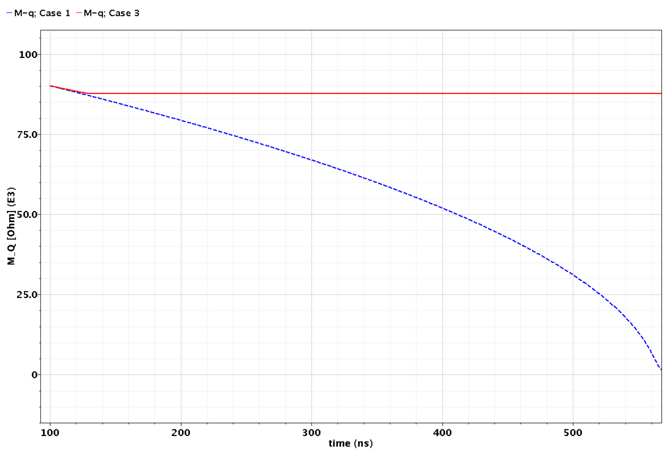


Figure 13. Memristance of *q* when applying an IMPLY logic gate for cases 1 (dashed line) and 3 (solid line) for a memristor with threshold model (current threshold is 7 µA).

TABLE 3. WRITE TIME AND STATE DRIFT FOR DIFFERENT VALUES OF V_{SET} AND R_G. ALL VALUES SATISFY (6) AND (8). V_{COND} IS SET TO 0.5 V.

V_{SET} [V]	R_G [kΩ]	T [µsec]	State Drift [% R_{OFF}]
1	5	0.47	2.44
0.8	5	0.592	~ 0
1.5	5	0.31	6
1	3.5	0.453	2.53
1	15	0.579	2.15

V. CONCLUSIONS

The logic design of a memristor-based IMPLY logic gate is presented. Investigating and characterizing the behavior of a memristor and IMPLY logic gate reveals several design limitations and considerations. The IMPLY logic gate trades off performance (write time) with robustness (internal state drift). This tradeoff requires the circuit to be occasionally refreshed.

Several heuristics for designing IMPLY logic gates with memristors are proposed and organized into a design procedure. This design procedure considers the influences and tradeoffs among the different input cases, initial conditions, and circuit parameters of the memristor.

A design example based on the proposed design procedure is presented and compared with simulation. It is shown that the widely used linear ion drift model is incompatible with the IMPLY logic gate, since under this model, the state drift phenomenon is excessively high. To accurately characterize the IMPLY logic gate operation, a highly non-linear memristor model needs to be used; or alternatively, a device with a threshold. The proposed design procedure is the first step in the development of a general design methodology for logic gates based on memristors.

REFERENCES

[1] L. O. Chua, “Memristor – the Missing Circuit Element,” *IEEE Transactions on Circuit Theory*, Vol. 18, No. 5, pp. 507-519, September 1971.

[2] D. B. Strukov, G. S.Snider, D. R. Stewart, and R. S. Williams, "The Missing Memristor Found,” *Nature*, Vol. 453, pp. 80-83, May 2008.

[3] Y. Ho, G. M. Huang, P. Li, "Nonvolatile Memristor Memory: Device Characteristics and Design Implications," *Proceedings of the IEEE International Conference on Computer-Aided Design*, pp. 485-490, November 2009.

[4] A. Afifi, A. Ayatollahi, and F. Raissi, "Implementation of Biologically Plausible Spiking Neural Network Models on the Memristor Crossbar-based CMOS/Nano Circuits," *Proceedings of the European Conference on Circuit Theory and Design*, pp. 563- 566, August 2009.

[5] Y. V. Pershin and M. Di Ventra, "Practical Approach to Programmable Analog Circuits with Memristors," *IEEE Transactions on Circuits and Systems I: Regular Papers*, Vol. 57, No. 8, pp. 1857-1864, August 2010.

[6] D. B. Strukov and K. K. Likharev, "CMOL FPGA: a Reconfigurable Architecture for Hybrid Digital Circuits with Two-Terminal Nanodevices," *Nanotechnology*, Vol. 16, No. 6, pp. 888-900, June 2005.

[7] G. S. Snider and R. S. Williams, "Nano/CMOS Architectures Using a Field-Programmable Nanowire Interconnect," *Nanotechnology*, Vol. 18, No. 3, 035204, January 2007.

[8] G. S. Rose and M. R. Stan, "A Programmable Majority Logic Array Using Molecular Scale Electronics," *IEEE Transactions on Circuits and Systems I: Regular Papers*, Vol. 54, No. 11, pp. 2380-2390, November 2007.

[9] G. Snider, "Computing with Hysteretic Resistor Crossbars," *Applied Physics A: Materials Science and Processing*, Vol. 80, No. 6, pp. 1165-1172, March 2005.

[10] J. Borghetti, G. S. Snider, P. J. Kuekes, J. J. Yang, D. R. Stewart, and R. S. Williams, "Memristive Switches Enable 'Stateful' Logic Operations via Material Implication," *Nature*, Vol. 464, pp. 873-876, April 2010.

[11] N. Weste and D. Harris, *CMOS VLSI Design: A Circuits and Systems Perspective*, Addison Wesley, 2010.

[12] E. Lehtonen and M. Laiho, "Stateful Implication Logic with Memristors," *Proceedings of the IEEE/ACM International Symposium on Nanoscale Architectures*, pp. 33-36, July 2009.

[13] E. Lehtonen, J. H. Poikonen, and M. Laiho, "Two Memristors Suffice to Compute All Boolean Functions," *Electronics Letters*, Vol. 46, No. 3, pp. 239-240, February 2010.

[14] K. Eshraghian, K. R. Cho, O. Kavehei, S. K. Kang, D. Abbot, and S. M. S. Kang, "Memristor MOS Content Addressable Memory (MCAM): Hybrid Architecture for Future High Performance Search Engines," *IEEE Transactions on Very Large Scale Integrated Systems*, in press.

[15] Z. Biolek, D. Biolek, and V. Biolkova, "Spice Model of Memristor with Nonlinear Dopant Drift," *Radioengineering*, Vol. 18, No. 2, Part 2, pp. 210-214, June 2009.

[16] S. Kvatinsky, E. G. Friedman, A. Kolodny and U. C. Weiser, "TEAM: ThrEshold Adaptive Memristor Model," unpublished.