Control Circuit Overhead for Stateful Memristor Logic

Xuan Hu, Michael J. Schultis, Matthew Kramer, Archit Bagla, Akshay Shetty and Joseph S. Friedman
Department of Electrical and Computer Engineering, The University of Texas at Dallas, Richardson, TX 75080 USA

***Abstract*—Memristors are considered as a potential replacement of CMOS transistors in light of their low switching energy, high density and non-volatility. These two-terminal devices have been shown to be versatile and efficient for implementing logical, neuromorphic, and in-memory computing systems. However, memristor-based computing circuits require complex sequential operations that are controlled by overhead circuity, and these control circuits have not been considered in previous efficiency analyses of memristor logic. Here we thoroughly analyze the overhead circuitry for memristor logic and evaluate the actual area, power, and delay consumptions for complete memristor logic systems. This evaluation includes a horizontal comparison between the original IMPLY-based memristor logic to pure-CMOS logic in implementing standard logic gates and a one-bit full adder. The results show that when the overhead control circuitry is considered, memristor logic is ten orders of magnitude less efficient than conventional CMOS logic.**

Keywords—memristors, memristive systems, non-volatile logic computing, beyond CMOS computing, memristor control circuit

I. Introduction

Memristors have been explored deeply for next-generation logical, neuromorphic, and in-memory computing (IMC) systems due to their potential for non-volatility, low switching energy, and high density. However, the non-volatility and the two-terminal connectivity of memristors necessitate complex signaling to switch the device state. To properly deliver these input stimuli, complex CMOS driving circuits are needed that consume enormous quantities of energy. Furthermore, memristor logic requires clocked sequential steps to perform most basic logic functions, rather than a single combinational operation as in CMOS logic [1]. This sequential control mechanism needs to be implemented in additional circuitry in order for the system to store and execute these sequential operations. This requirement of excessive CMOS control circuitry poses a serious threat that must be considered in order to enable realistic predictions of the efficiency of this technology [2].

II. Background: Memristors & Stateful Logic

Fig. 1(a) shows the symbol of a memristor. A positive voltage applied to the "+" terminal relative to the "–" terminal that exceeds a specific threshold voltage decreases the resistance to R_{ON}, while a negative voltage applied to the "+" terminal relative to the "–" terminal that exceeds a specific threshold voltage increases the resistance to R_{OFF} [3]. Stateful memristor logic leverages this non-volatile threshold switching, and has been demonstrated both theoretically [1] and experimentally [4], [5].

Fig. 1(b) shows the schematic of a memristor-based logic gate that implements the material implication function (IMPLY), while its truth table is shown in Table I. This IMPLY gate is the most fundamental stateful memristor logic function, as it requires only two memristors and one operation step. Three different voltage levels are required to execute the operation correctly: conditional voltage V_{COND}, set voltage V_{SET}, and reset voltage V_{RESET}. The memristor switches states from high resistance (0) to low resistance (1) when the V_{SET} voltage is applied across its terminals from + to −. The V_{COND} voltage is of less magnitude than V_{SET}, such that applying V_{COND} or (V_{SET} - V_{COND}) across a memristor does not change the memristor from high resistance (0) to low resistance (1). Furthermore, since memristors are non-volatile, a reset operation is needed to change the memristor from low resistance (1) to high resistance (0) by applying V_{RESET} across the memristor. Thus, there are four possible connectivity configurations for the + node of each memristor in the IMPLY-based memristor logic: V_{SET}, V_{COND}, V_{RESET}, and floating [2].

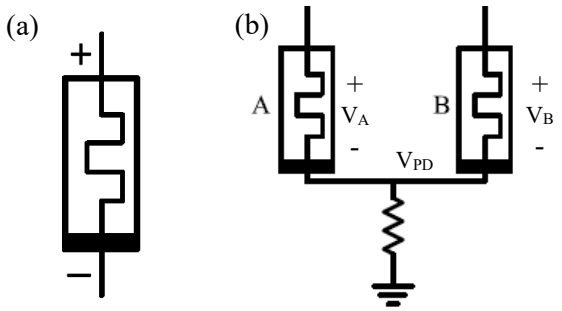

Fig. 1. (a) Memristor symbol and (b) device-level schematic of a memristor-based material implication gate.

TABLE I. IMPLY Function Truth Table And Node Voltages

	Input		During Operation		Output
Case	A	B	V_{PD}	V_B	B' = A → B
1	0	0	GND	V_{SET}	1
2	0	1	V_{SET}	0	1
3	1	0	V_{COND}	V_{SET}-V_{COND}	0
4	1	1	$V_{SET} < V_{PD} < V_{COND}$	0	1

It should be noted that in addition to IMPLY-based logic, numerous other memristor logic approaches have been proposed [6]-[8]. These other approaches, however, require even more CMOS control circuitry than IMPLY-based stateful memristor logic, thereby necessitating even greater area and energy overhead.

III. Control Circuit Overhead Analysis

Figure 2 shows a system architecture of the control circuit for IMPLY-based memristor logic [2]. For each memristor, the V_{SET}, V_{COND}, and V_{RESET} must all be accessible; therefore, each memristor requires three power transistors to deliver the proper

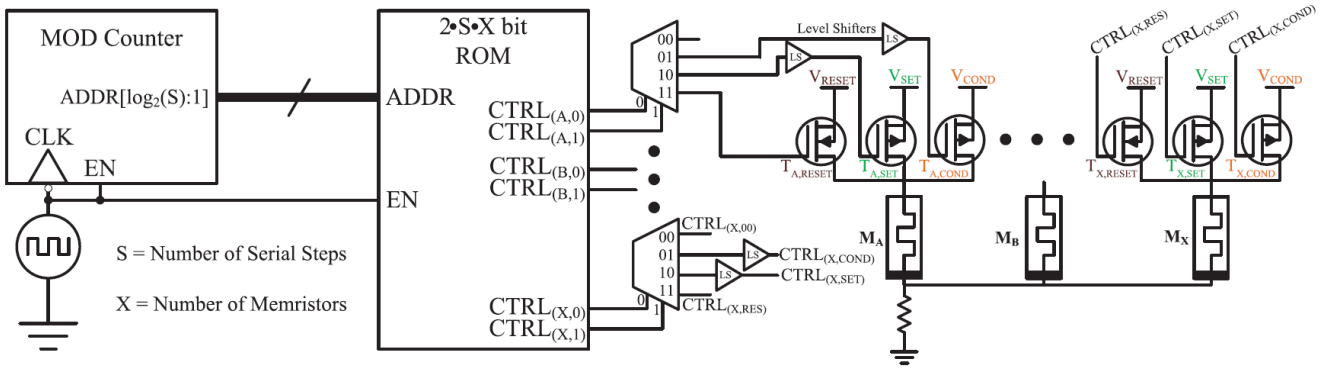

Fig. 2. Block diagram for memory-based, clocked control scheme that scales with memristor requirements.

TABLE II. NUMBER OF DEVICES REQUIRED TO PERFORM STANDARD FUNCTIONS WITH SUPPORTING TRANSISTOR COUNT FOR MEMRISTORS

Function	CMOS Only	Memristor + CMOS
IMPLY	6T	2M + 104T
AND	6T	4M + 292T
OR	6T	4M + 285T
NAND	4T	3M + 231T
NOR	4T	3M + 196T
XOR	8T	5M + 473T
XNOR	8T	4M + 410T
Full Adder	28T	6M + 676T

TABLE III ENERGY-DELAY PRODUCT FOR MEMRISTORS AND CMOS

Function	Memristor EDP (J·s)	CMOS EDP (J·s)
NAND	$2.58 * 10^{-13}$	$1.87 * 10^{-24}$
AND	$3.95 * 10^{-13}$	$3.59 * 10^{-24}$
NOR	$1.22 * 10^{-13}$	$2.53 * 10^{-24}$
OR	$3.80 * 10^{-13}$	$4.46 * 10^{-24}$
XOR	$7.25 * 10^{-12}$	$9.31 * 10^{-24}$
XNOR	$6.04 * 10^{-12}$	$9.15 * 10^{-24}$
Full Adder	$1.93 * 10^{-11}$	$3.32 * 10^{-23}$

driving voltage. A system clock signal increments a counter circuit that generates the address signal that is provided to a ROM which contains the serial step information to implement the desired function. The instructions from the ROM output are used to drive a set of decoders that control the CMOS power transistors driving each memristor. This block diagram is a fully-scalable control block that can be used to control any number of memristors with any number of steps for the computation.

Table II summarizes the number of devices required to perform standard logic functions in conventional CMOS logic and memristor logic with supporting transistors [2]. Though the memristor count (M) in IMPLY-based logic is less than the number of the transistors (T) in the pure-CMOS logic, when the overhead circuitry is taken into account, the memristor logic requires hundreds of extra CMOS transistors to aid in implementing the same logic functions. Beyond the extra area and energy cost of using CMOS transistors in the overhead control circuitry, the sequential operation mechanism of the memristor logic leads to a longer delay. Table III shows the energy-delay product (EDP) of performing the same logic functions in both stateful memristor logic and pure-CMOS logic, demonstrating that *stateful memristor logic has an EDP ten orders-of-magnitude higher than conventional CMOS implementations* [2].

IV. CONCLUSIONS

Memristors have been explored as potential replacements for CMOS transistors in the beyond-CMOS era in light of their non-volatility, low switching energy, and high density. When evaluating the potential of memristors in next-generation computing systems, it is important to include the required overhead CMOS control circuity in the evaluation of the memristor-based logic systems. Here we demonstrate that the control circuitry required for stateful memristor logic has enormous overhead costs, leading to an EDP more than ten orders-of-magnitude greater than conventional CMOS logic implementations. Therefore, though stateful memristor logic may have utility within a non-von Neumann computing architecture that makes efficient use of the memristor non-volatility, the prospects for stateful memristor logic within a conventional computer architecture are severely limited by the overhead of the control circuit.

REFERENCES

[1] S. Kvatinsky, G. Satat, N. Wald, E. G. Friedman, A. Kolodny, and U. C. Weiser, "Memristor-based material implication (IMPLY) logic: Design principles and methodologies," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 22, no. 10, pp. 2054–2066, Oct. 2014.

[2] X. Hu, M. J. Schultis, M. Kramer, A. Bagla, A. Shetty, and J. S. Friedman, "Overhead Requirements for Stateful Memristor Logic," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 66, no. 1, pp. 263-273, 2019

[3] S. Kvatinsky, M. Ramadan, E. G. Friedman, and A. Kolodny, "VTEAM: A general model for voltage-controlled memristors," *IEEE Trans. Circuits Syst., II, Exp. Briefs*, vol. 62, no. 8, pp. 786–790, Aug. 2015.

[4] 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, no. 7290, pp. 873–876, Apr. 2010.

[5] 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.

[6] S. Kvatinsky, N. Wald, G. Satat, A. Kolodny, U. C. Weiser, and E. G. Friedman, "MRL—Memristor ratioed logic," in *Proc. Int. Workshop Cellular Nanosc. Netw. Their Appl.*, 2012, pp. 1–6.

[7] S. Kvatinsky et al., "MAGIC—Memristor-aided logic," *IEEE Trans. Circuits Syst., II, Exp. Briefs*, vol. 61, no. 11, pp. 895–899, Nov. 2014.

[8] E. Linn, R. Rosezin, C. Kügeler, and R. Waser, "Complementary resistive switches for passive nanocrossbar memories," *Nature Mater.*, vol. 9, no. 5, pp. 403–406, May 2010.