Received 10 March 2020; accepted 6 April 2020. Date of publication 14 April 2020; date of current version 7 August 2020.
The review of this article was arranged by Editor L. Lukasiak.

Digital Object Identifier 10.1109/JEDS.2020.2987402

Smart Logic-in-Memory Architecture for Low-Power Non-Von Neumann Computing

TOMMASO ZANOTTI (Graduate Student Member, IEEE), FRANCESCO MARIA PUGLISI (Member, IEEE), AND PAOLO PAVAN (Senior Member, IEEE)

Dipartimento di Ingegneria "Enzo Ferrari," Università di Modena e Reggio Emilia, 41125 Modena, Italy

CORRESPONDING AUTHOR: T. ZANOTTI (e-mail: tommaso.zanotti@unimore.it).

ABSTRACT Low-power smart devices are becoming pervasive in our world. Thus, relevant research efforts are directed to the development of innovative low power computing solutions that enable in-memory computations of logic-operations, thus avoiding the von Neumann bottleneck, i.e., the known showstopper of traditional computing architectures. Emerging non-volatile memory technologies, in particular Resistive Random Access memories, have been shown to be particularly suitable to implement logic-in-memory (LIM) circuits based on the material implication logic (IMPLY). However, RRAM devices non-idealities, logic state degradation, and a narrow design space limit the adoption of this logic scheme. In this work, we use a physics-based compact model to study an innovative smart IMPLY (SIMPLY) logic scheme which exploits the peripheral circuitry embedded in ordinary IMPLY architectures to solve the mentioned reliability issues, drastically reducing the energy consumption and setting clear design strategies. We then use SIMPLY to implement a 1-bit full adder and compare the results with other LIM solutions proposed in the literature.

INDEX TERMS Compact model, full adder, IMPLY, logic-in-memory, RRAM.

I. INTRODUCTION

The Internet of Things (IoT) is promoting the spreading of battery powered smart devices in many sectors, from the industry, to improve the efficiency of industrial processing, to the healthcare, to monitor patients from distance, and in many others. This trend is posing a particular interest in the research of innovative solutions that enable the computation of logic operations directly inside the memory [1], [2], with the aim of bypassing the strongest limitation affecting traditional architectures, i.e., the data transfer between the memory and the CPU, also known as the von Neumann bottleneck [3]. Among emerging technologies, Resistive Random Access Memories (RRAM) are indeed a promising candidate for Logic-in-Memory (LIM) applications, as they are non-volatile storing elements which at the same time can also act as computing element in logic gates [4], [5]. In addition, RRAM devices offer very fast-switching [6], small footprint [4], [7], and can be integrated in the Back-End-Of-Line (BEOL) [4], [8] of integrated circuits, also in dense crossbar array structures. A specific scheme, based on the material implication logic (IMPLY) has been identified as an effective solution for fast, low-power LIM architectures. Its functionality has been explored in many simulation works [9]–[11], and recently experimentally demonstrated [5], [12]. Although encouraging, this scheme is affected by many severe issues such as the logic state degradation [9]–[10], [13]–[14] and the strong sensitivity to driving voltage variations [13]–[14], which can prevent the correct circuit functionality if not considered during the design phase. Still, these issues are often disregarded, and simulations are typically performed using simplified device models [9], [15] that can reproduce the quasi-static DC IV characteristic but have a questionable accuracy when considering ultra-fast pulses, as the temperature dynamics becomes a critical factor. Thus, physics-based compact models are essential when studying this kind of circuits [13]. Moreover, previous works mainly focused on the circuit short-term functionality, but only a few studies have analyzed the circuit reliability and the long-term functionality considering the logic state degradation. This paper extends the conference paper in [14], where we used a physics-based compact model to propose and evaluate through simulations the feasibility

of an innovative RRAM-based smart IMPLY scheme, i.e., SIMPLY, which removes the issues affecting the ordinary IMPLY scheme. SIMPLY i) virtually solves the problem of logic state degradation, ii) removes the strict tradeoff on the driving voltage choice, iii) considerably reduces the energy consumption without adding relevant area and complexity overheads. Here, we further investigate the performance of SIMPLY by including in the design the necessary peripheral circuitry, i.e., a small area comparator. We then simulate the performance of a 1-bit full adder using the SIMPLY and IMPLY schemes on a RRAM array and compare the results with other state-of-the-art LIM solutions.

II. RRAM COMPACT MODEL FOR RELIABLE CIRCUIT DESIGNS

Studying the reliability of RRAM-based logic circuits to obtain accurate estimates of the circuit long-term performance requires dependable RRAM compact models. Such compact models need to accurately reproduce the dynamic behavior and to account for device non-idealities, such as cycle-to-cycle and device-to-device variability, self-heating, and Random Telegraph Noise (RTN) [13]. In this work we use a physics-based RRAM compact model [10], [16], implemented in Verilog-A and available on nanohub [17]. As summarized in the sketches in Fig. 1 (c)-(d)-(e), the total device resistance consists of the sum of two components, a metallic-like conductive filament (CF) and a dielectric barrier. Set and reset operations, that affect the resistance of the two components and result in a state change of the device, are reproduced by considering the field- and temperature- driven bond breaking and related defect generation during set, and the field- and temperature- assisted oxygen ions drift and recombination with vacancies during reset [10], [16]. Since the barrier and the CF differ in chemical composition, we avoid introducing the simplifying assumption of a spatially homogeneous device temperature. Instead, we consider two separate temperatures for the CF and dielectric barrier components (T_{cf} and T_b in Fig. 1 (e), respectively), thus strengthening the physical soundness and accuracy of the model. These temperatures are modeled dynamically including the effect of thermal capacitance, therefore enabling accurate predictions when performing simulations using very-short pulses (Fig. 1 (b)). Cycle-to-cycle and device-to-device variability in both resistive states are included by adding appropriate random noise sources on the dielectric barrier during reset, and on the CF cross-section during set. In addition the model reproduces multilevel Random Telegraph Noise (RTN) [16], this being a distinctive feature that is necessary when assessing the reliability of the device read operation. In this work we calibrated the model on experimental data of a multilayer TiO_x/HfO_x RRAM from [18]. As shown in Fig. 1 (a)-(b), the model correctly reproduces both DC I-V and ultra-fast pulsed characteristics (i.e., 10 ns pulses), with a single set of parameters.

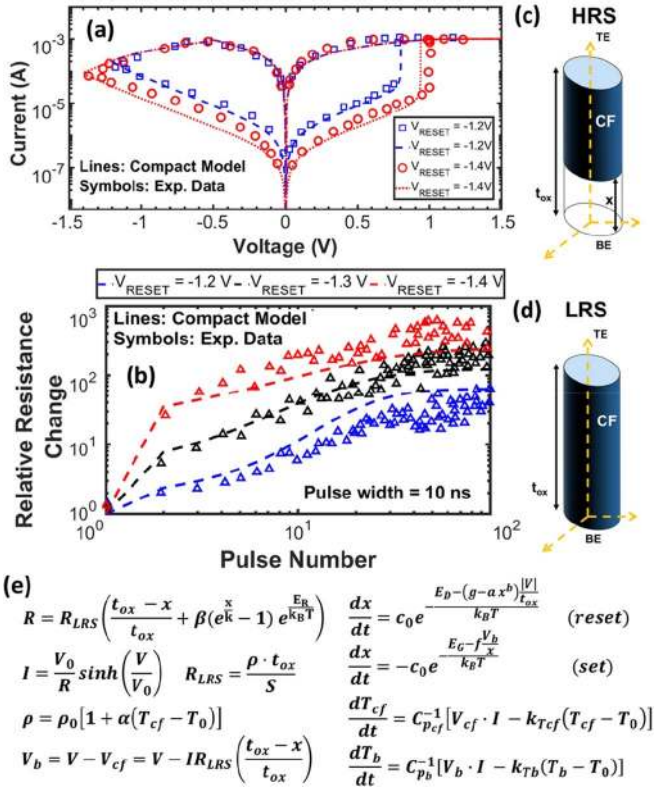


FIGURE 1. (a) Simulated (lines) and experimental (symbols) DC I-V curves of a multilayer TiOx/HfOx RRAM device during set and reset operations in different reset conditions (i.e., $V_{RESET} = -1.2$ V, and -1.4 V, with different colors). Data from [18]. (b) Simulated (lines) and experimental (symbols) pulsed reset curves using a train of 10 ns pulses with different voltages (i.e., $V_{RESET} = -1.2$ V, -1.3 V, and -1.4 V, with different colors). Data from [18]. (c-d) Sketch of the compact representation of a device in HRS (c) and LRS (d). (e) The compact model equations, where R is the total device resistance, R_{LRS} the value of R in LRS, t_{ox} the oxide thickness (12 nm in this case), x the barrier thickness, k the typical tunneling length, T the device temperature, T_0 the ambient temperature, k_B the Boltzmann constant, I the current, V the applied voltage, S the filament cross-section, ρ its resistivity. E_R, E_D, E_G, g, f, c_0, a, b, β, k_{Tcf}, k_{Tb}, C_{pcf} and C_{pb} are material-related constants for defect diffusion and generation, and for thermal dissipation [10], [16]–[17].

III. IMPLY LOGIC RRAM IMPLEMENTATION

Although theorized at the beginning of the 20th century by Russell and Whitehead [19], the IMPLY logic has never found application in electronic circuits, as it is less prone to be implemented with transistors than the Boolean operations AND, OR, and NOT. It was only recently when Borghetti *et al.* [5] experimentally demonstrated the feasibility of RRAM-based LIM circuits, that this logic scheme has revived the interest of the electronic community. The IMPLY is a stateful logic, meaning that a RRAM device functions both as a logic gate and as a memory element. Together with the FALSE (i.e., an operation resulting always in a logic-0), the IMPLY enables the computation of any logic operation [20] by decomposing it in a sequence of IMPLY and FALSE operations on an appropriate number of devices (the number of input devices plus at least 2 additional devices [21]). The COPY operation (which copies

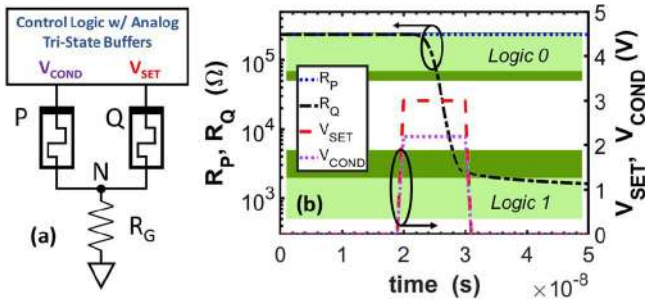


FIGURE 2. Schematic representation of the ordinary RRAM-based IMPLY. (b) Simulated resistance of P (R_P, blue dotted line) and Q (R_Q, black dashed line) over time during the execution of the IMPLY operation, along with the pulses applied to the top electrodes of the two RRAMs (i.e., V_{SET} and V_{COND}, dashed red and dotted violet lines, respectively). P and Q are initially set to logic 0. Q switches to logic 1, as per the truth table in Table 1. The bands of the resistance intervals defining the logic states are also reported (green bands).

TABLE 1. Truth table and requirements for the ordinary IMPLY gate functionality (*) and optimized degradation (#).

Input Combination		Output (Q')	Requirements
P = 0	Q = 0	Q' = 1	*High V_{SET} *#Low V_{COND}
P = 0	Q = 1	Q' = 1	#Low $\lvert V_{COND} - V_{SET} \rvert$
P = 1	Q = 0	Q' = 0	#Low $\lvert V_{COND} - V_{SET} \rvert$
P = 1	Q = 1	Q' = 1	-

the content of one memory element to another) can be implemented with IMPLY and FALSE, enabling cascaded operations [12].

A. WORKING PRINCIPLES

A single IMPLY logic gate can be implemented with two RRAMs with their bottom electrodes (BEs) connected to ground through a common resistor R_G, see Fig. 2(a). Logic bits are associated to the resistive states of RRAM devices: a device in high resistive state (HRS) is a logic-0, while a device in low resistive state (LRS) is a logic-1. To perform the P-IMPLY-Q operation, the control logic applies two simultaneous positive voltage pulses at the top electrodes of the devices P and Q, as shown in Fig. 2(b). These two pulses are sized so that the state of P never changes, while the state of Q changes according to the operation truth table (Table 1) [5], [9], [10], [13]. Thus, the state of Q after the operation execution (Q') is the result of the operation. The FALSE operation consists in the application of a negative voltage pulse to a single device to restore its HRS (logic-0). In the rest of the paper, we use the following circuit parameters $R_G = 1k\Omega$, $V_{SET} = 2.15V$ (where V_{SET} is the voltage at the TE of the device Q), $V_{COND} = 1.7V$ (where V_{COND} is the voltage at the TE of the device P), $V_{FALSE} = -1.45V$, with all the pulses having a 20ns period with 50% duty cycle. These parameters were derived in [13] and represent a good compromise between energy consumption and enhanced circuit reliability. Resistive state variability strongly affects the reliability of the IMPLY gate. Therefore, to map the logic states on resistive states we consider resistance intervals instead of a single threshold as described in previous works [5], [9], [12], [21]. We defined a logic-0 as a resistance in the range 50kΩ to 300kΩ, and a logic-1 as a resistance in the range 500Ω to 5kΩ, see color bands in Fig. 2 (b). These intervals were defined by considering the variability displayed by the devices in operating conditions, which leads to a $R_{LRS,MIN} = 500\Omega$ and $R_{LRS,MAX} = 2$ kΩ, $R_{HRS,MIN} = 70$ kΩ and $R_{HRS,MAX} = 230$ kΩ, which correspond to the light green bands reported in Fig. 2b. Both HRS and LRS ranges were enlarged to introduce tolerance margins (dark green bands) to account for the presence of RTN.

B. ISSUES AND LIMITATIONS

Although these circuits display low energy consumption and high integration density, they suffer from several issues that limit their reliability and increase their design complexity by introducing strict requirements. First, only a narrow subspace of V_{SET} and V_{COND} pairs results in the correct circuit functionality [13]. Considering the P-IMPLY-Q operation, a too high V_{COND} can cause the switching of device P, which instead must always retain its state, while a too low V_{COND} cannot inhibit the switching of Q when P = 1 and Q = 0. Instead, V_{SET} should be high enough to switch Q when P = Q = 0 and at the same time low enough to prevent the switching of Q when P = 1 and Q = 0. Secondly, at each IMPLY execution, the resistance of a device in the HRS that is supposed to hold a logic-0 decreases, and after a limited number of cycles determines bit corruption (Fig. 3) [9]–[10], [13]–[14]. This requires the introduction of a periodic energy-hungry memory refresh (a global COPY followed by a global FALSE). In particular, the state degradation happens on the device P when P = Q = 0, and on the device Q when P = 1 and Q = 0, since a positive voltage drop, even if lower than the switching voltage, drops across the devices P and Q, respectively, and determines a partial set event. The state drift strongly depends on the driving voltage values which are affected by a strict tradeoff, as highlighted in Table 1. In fact, lowering V_{COND} (V_{SET}) improves the degradation when P = Q = 0 (P = 1 and Q = 0) but worsens it when P = 1 and Q = 0 (P = Q = 0). The analysis of degradation is non-trivial and the effects may be overlooked when using simplified general-purpose models which consider a hard threshold to activate the state variable dynamics [9]. Moreover, variability must be simultaneously considered, as the state degradation strongly depends on the actual device resistances, as depicted in Fig. 3. The above issues strongly limit the circuit reliability, and introduce the requirement of a precise control of the driving voltages up to the tens of mV [13] which is hard to achieve in hardware: voltage drops across the parasitic line resistances and noise at voltage sources can easily shift the gate operating point into a high degradation region or even outside the allowed space.

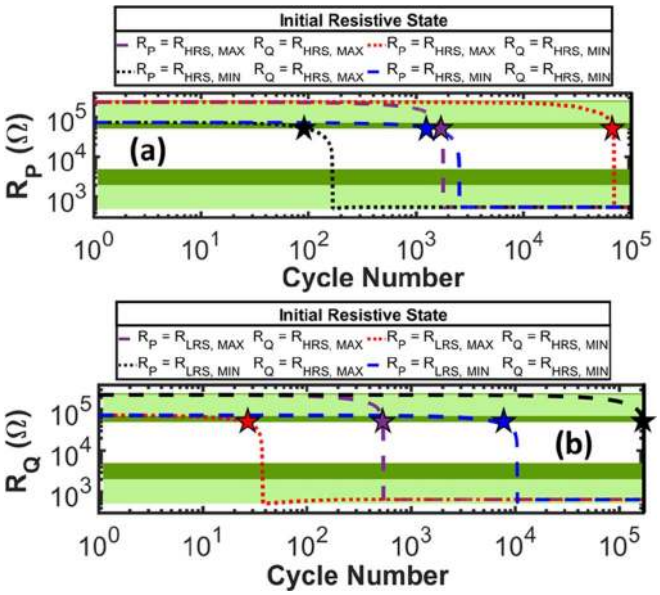


FIGURE 3. Degradation of R_P over time during the repeated execution of the IMPLY operation with P = Q = 0. Four corner cases are considered to account for variability (R_P is initialized either at $R_{HRS,MIN}$ or at $R_{HRS,MAX}$ and R_Q is kept fixed either at $R_{HRS,MIN}$ or at $R_{HRS,MAX}$). The degradation depends on the initial values of R_Q and R_P, resulting in errors (star symbols) potentially after ∼100 cycles (black star). (b) Degradation of R_Q over time during the repeated execution of the IMPLY operation with P = 1 and Q = 0. Four corner cases are considered to account for variability (R_P is initialized either at $R_{LRS,MIN}$ or at $R_{LRS,MAX}$ and R_Q is initialized either at $R_{HRS,MIN}$ or at $R_{HRS,MAX}$). In the worst case, the degradation of R_Q results in errors after ∼30 cycles (red star).

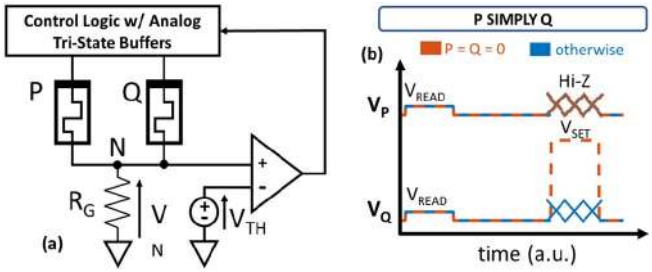


FIGURE 4. Schematic of SIMPLY architecture. (b) Schematic diagram of the voltage pulses delivered to the top electrodes of P (V_P) and Q (V_Q) over time during the execution of the SIMPLY operation when the comparator detects the condition P = Q = 0 (dashed orange lines) and in all other cases (blue lines).

IV. SIMPLY OPERATION SCHEME

To solve the issues of the ordinary IMPLY, we proposed a novel LIM smart scheme, i.e., SIMPLY (smart IMPLY) [14]. This scheme, without introducing additional complexity in the traditional IMPLY scheme, is capable of breaking the tradeoff between V_{SET} and V_{COND}. This virtually solves the problem of state degradation and its dependence on the resistive state values, while reducing energy consumption and setting clear design tuning parameters.

A. WORKING PRINCIPLES

The idea behind SIMPLY is based on the observation that, when performing the IMPLY operation, the state of Q changes only when the input combination is P = Q = 0. In all the other cases, Q retains its initial state. So, it is possible to distinguish this input combination from all the others, by simply applying two simultaneous small voltage pulses ($V_{READ} \approx 100$mV) at the top electrodes (TEs) of P and Q and comparing the voltage at node N (V_N) with a threshold V_{TH}, which must lie between the maximum value of V_N when P = Q = 0 and the minimum V_N when P ≠ Q. The difference between the two latter quantities is labeled read margin (RM). The output of the comparator is fed back to the control logic (Fig. 4(a)) which then activates the analog tri-state buffers and pulses V_{SET} on the TE of Q if $V_N < V_{TH}$, or keeps the buffers in high impedance otherwise, as shown in Fig. 4 (b). Thus, V_{COND} is no more necessary and the tradeoff eliminated, so that V_{SET} must just be high enough to switch a device considering the maximum HRS resistance due to the device variability introduced by the reset operation. In addition, in three out of four cases of the truth table only small V_{READ} pulses are applied to the devices, resulting in a considerable energy reduction. To correctly perform a SIMPLY operation, only a large enough RM must be provided. We performed simulations including the effect of variability and RTN and verified that, even with a V_{READ} of 100mV, an RM of at least 40mV is guaranteed, as illustrated in Fig. 5. In addition, clear design strategies can be defined, as the RM can be improved by increasing the power consumption, choosing either higher V_{READ} (Fig. 5) or higher $|V_{FALSE}|$ to increase the R_{HRS}/R_{LRS} ratio. It is important to note, that the SIMPLY scheme only introduces minimal changes in the control logic, while the analog tri-state buffers together with the comparator and the V_{READ} are already embedded in the IMPLY architecture peripheral circuitry to read the state of a device [22]. To introduce the benefits of SIMPLY, the comparator must have a small footprint and dissipate low energy (i.e., comparable to the energy of the read operation). In Fig. 6, we propose a sense amplifier design, implemented in standard 45nm CMOS technology [26], that satisfies the above requirements.

V. IMPLY VS SIMPLY

In this section, we compare the performance of the IMPLY and SIMPLY operations in terms of energy and reliability, based on the results of extensive circuit simulations of the single operation alone. In addition, we also compare the performance of the two architectures when performing a 1-bit full addition computed on a RRAM linear array. For the IMPLY we used the parameters defined in Section III-A, while for the SIMPLY we used $V_{SET} = 2.15$V, $V_{READ} = 200$mV, $V_{FALSE} = -1.45$V, $V_{DD} = 2$V (which is the comparator power supply), $R_G = 1$kΩ, and the same timing as in the IMPLY operation. This means that $T_{SIMPLY} = 2 \cdot T_{IMPLY}$, because in SIMPLY one T_{IMPLY} is needed to perform the read and compare step, and another to possibly set the device, see Fig. 4(b).

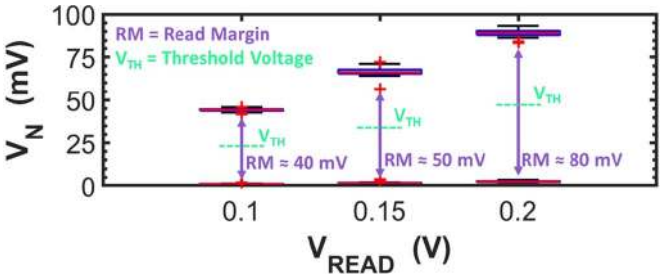


FIGURE 5. Distribution of V_N calculated at different V_{READ}, including the effect of variability and RTN over 50 cycles. The upper and lower boxes represent the cases $P \neq Q$ and $P = Q = 0$, respectively, showing the medians (red lines), the 25^{th} to 75^{th} percentiles extension (blue boxes), and the most extreme data points (black whiskers). The red crosses represent the worst-case fluctuations due to RTN. The read margins (RMs – violet arrows) and associated threshold voltages (V_{TH}s – ocean green dashed lines) for the comparator are shown.

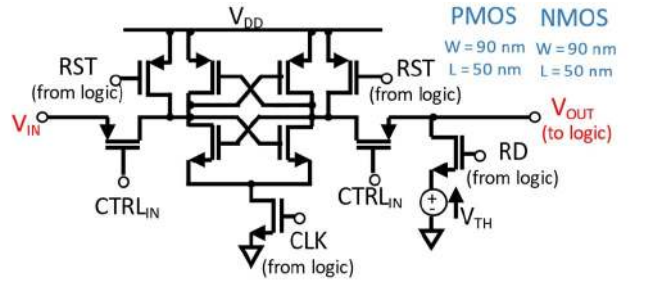
TABLE 2. IMPLY vs SIMPLY energy consumption comparison.

Input Combination		Energy Min- Avg.-Max (Ordinary IMPLY)	Energy Min- Avg.-Max (SIMPLY)
P = 0	Q = 0	26.3 – **29.1** – 30.7 **pJ**	27.5 – **28.9** – 30.6 **pJ**
P = 0	Q = 1	31.6 – **35.9** – 37.8 **pJ**	135 – **221** – 269 **fJ**
P = 1	Q = 0	19.3 – **20.8** – 22.1 **pJ**	110 – **213** – 252 **fJ**
P = 1	Q = 1	26.7– **28.5** – 30.3 **pJ**	245 – **286** – 307 **fJ**
FALSE		4.0 – **8.2** – 12.1 **pJ**	

A. SINGLE OPERATION COMPARISON

As described in Section III-B, one of the major limitations of the IMPLY operation is the problem of logic state degradation. In fact, the results of the simulation of the IMPLY considering eight corner cases for the initial devices' resistances, shown in Fig. 3, reveal that in the worst-case a device can withstand no more than 30 cycles, therefore requiring frequent memory refresh which highly degrades circuits performances. On the contrary, in SIMPLY the logic state degradation is dramatically slowed, as it is determined only by a small V_{READ} voltage instead of a much larger one caused by the V_{SET}, V_{COND} pair. The improvement is clearly depicted in Fig. 7, where no relevant state drift can be seen up to $4.5 \cdot 10^6$ operation repetitions when considering the same two worst-cases of the IMPLY. Also, the energy per operation considerably drops in three out of four cases when only the V_{READ} voltage pulses are applied to the devices. This is shown in Table 2 where the energy per operation is reported for each input combination, obtained by including the effects of variability and RTN and repeating the simulation 20 times. The energy per comparison of the sense amplifier is not a strong limitation as it is comparable to the energy dissipated during the read operation. This is shown in Fig. 6 where the simulation results are reported considering a temperature range from 0°C up to 90°C, and including the effects of variability of the input RRAM devices.

To evaluate the improvement brought by SIMPLY, more complex operations should be considered since the sequence of individual IMPLY and FALSE operations determines the energy improvement. Thus, in the following section we present the simulation results of a 1-bit full adder.



	T = 273 (K)	T = 300 (K)	T = 363 (K)
EPO (fJ) (min-avg-max)	77 – 135 - 231	74 - 126 - 209	68 - 102 - 167

FIGURE 6. Sense Amplifier circuit used in this work as the comparator in the SIMPLY architecture. The circuit was implemented using a standard CMOS 45nm process [26]. The MOSFETs sizes are reported in blue. The table shows the minimum, average and maximum energy per operation for $V_{DD} = 2V$ and three different operating temperatures.

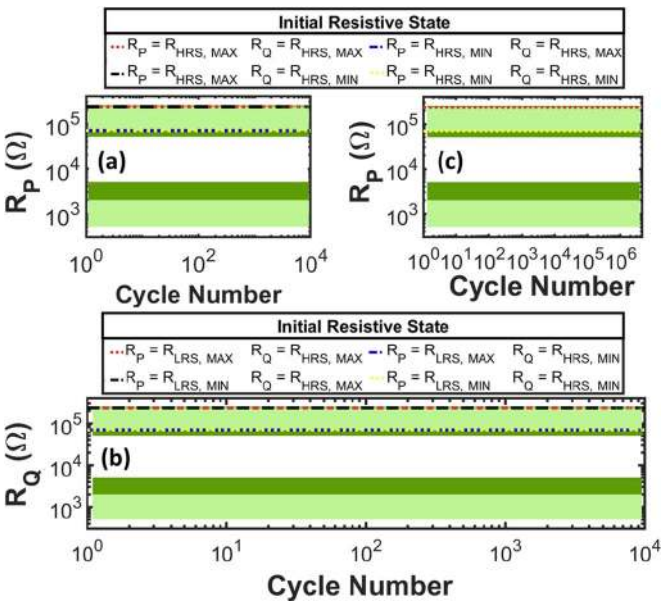


FIGURE 7. Benefits of using SIMPLY. (a) R_P vs. cycle number during the repeated execution of the IMPLY operation with $P = Q = 0$ and $V_{READ} = 200mV$. The same four corner cases as in Fig. 3(a) are considered to account for variability. No degradation is observed up to 10^4 cycles. (b) R_Q vs. cycle number during the repeated execution of the SIMPLY operation with $P = 1$ and $Q = 0$. The same four corner cases as in Fig. 3(b) are considered to account for variability. No degradation is observed up to 10^4 cycles. (c) No degradation up to $4 \cdot 5 \cdot 10^6$ cycles occurs for the two worst cases, i.e., $P = Q = 0$ with initial $R_P = R_Q = R_{HRS,MAX}$ (maximum voltage drop across P and Q) and $R_P = R_Q = R_{HRS,MIN}$ (maximum current on P and Q). In all cases, $V_{READ} = 200mV$.

B. 1-BIT FULL-ADDER COMPARISON

As shown in Table 5, which reviews the state-of-the-art of RRAM-based LIM 1-bit full adder implementations, there are several configurations of the number of devices and required computation steps that can be used to realize a

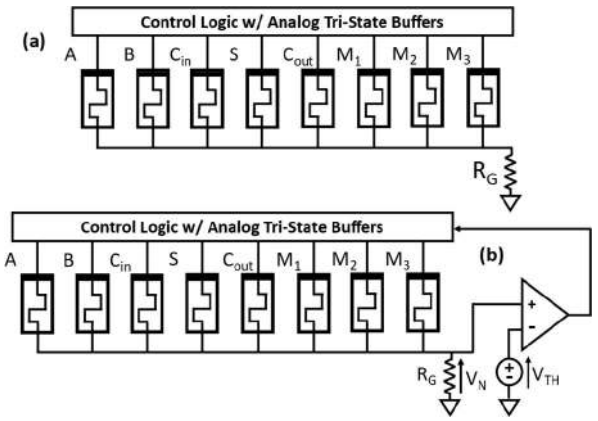


FIGURE 8. 1-bit full-adder implementation using (a) the IMPLY and (b) the SIMPLY architectures.

TABLE 3. Full adder energy per operation comparison: IMPLY vs SIMPLY.

Input Comb.			E_{FA} (min-avg-max)	E_{FA} (min-avg-max)
A	B	C_{in}	IMPLY	SIMPLY
0	0	0	508-**524**-538 pJ	186-**188**-191 pJ
0	0	1	504-**517**-523 pJ	189-**190**-193 pJ
0	1	0	506-**516**-527 pJ	164-**166**-169 pJ
0	1	1	488-**502**-514 pJ	144-**146**-149 pJ
1	0	0	508-**527**-545 pJ	186-**188**-191 pJ
1	0	1	503-**517**-539 pJ	166-**168**-171 pJ
1	1	0	506-**516**-527 pJ	164-**166**-169 pJ
1	1	1	509-**526**-540 pJ	164-**167**-169 pJ

TABLE 4. 1-bit full adder sequence of operations.

# Step	Operation	Equivalent Operation
1-5	$FALSE(M_1, M_2, M_3, S, C_{out})$	$M_1, M_2, M_3, S, C_{out} = 0$
6	$B\ IMP\ M_1$	$M_1 = \bar{B}$
7	$M_1\ IMP\ M_2$	$M_2 = B$
8	$A\ IMP\ M_2$	$M_2 = \bar{A} + B = \overline{\bar{A}\,B}$
9	$M_2\ IMP\ S$	$S = \bar{A}\,B$
10	$FALSE\ M_2$	$M_2 = 0$
11	$A\ IMP\ M_3$	$M_3 = \bar{A}$
12	$M_3\ IMP\ M_2$	$M_2 = A$
13	$B\ IMP\ M_2$	$M_2 = \bar{B} + A = \overline{A\bar{B}}$
14	$M_2\ IMP\ S$	$S = \bar{A}\,B + A\bar{B} = A \oplus B$
15	$B\ IMP\ M_3$	$M_3 = \bar{B} + \bar{A} = \overline{AB}$
16	$M_3\ IMP\ C_{out}$	$C_{out} = AB$
17	$FALSE\ M_3$	$M_3 = 0$
18	$S\ IMP\ M_3$	$M_3 = \overline{A \oplus B}$
19	$C_{in}\ IMP\ M_3$	$M_3 = \overline{C_{in}(A \oplus B)}$
20	$\boldsymbol{M_3\ IMP\ C_{out}}$	$\boldsymbol{C_{out} = AB + C_{in}(A \oplus B)}$
21	$FALSE\ M_1$	$M_1 = 0$
22	$C_{in}\ IMP\ M_1$	$M_1 = \overline{C_{in}}$
23	$M_1\ IMP\ S$	$S = \overline{\overline{C_{in}}(A \oplus B)}$
24	$FALSE\ M_2$	$M_2 = 0$
25	$M_2 = \overline{C_{in}}(A \oplus B)$	$M_2 = \overline{C_{in}}(A \oplus B)$
26	$M_3\ IMP\ M_2$	$M_2 = \overline{C_{in}}(A \oplus B) + C_{in}(A \oplus B)$
27	$FALSE\ S$	$S = 0$
28	$\boldsymbol{M_2\ IMP\ S}$	$\boldsymbol{S = C_{in} \oplus (A \oplus B)}$

1-bit full addition employing the IMPLY logic. Here, we use a RRAM linear array composed of 8 devices, as shown in Fig. 8 (a)-(b). Three devices hold the input bits (A, B, C_{in}), while the next two are used to store the final result of the computation (S, C_{out}), and the remaining three devices (M_1, M_2, M_3) are used to compute partial results. To perform an addition, we employ a scheme comprising 28 steps. These steps, listed in Table 4, include also the 5 FALSE required to initialize S, C_{out}, M_1, M_2, M_3, and offer the extra benefit of retaining the input values (i.e., the sequence of operations is engineered in such a way that the devices holding the input bits never change their logic state). This differs from most of the other works reported in Table 5, were the computation corrupts the values stored in the input devices. Therefore, if the stored value must be preserved, 3 additional COPY operations (realized with 6 FALSE and 6 IMPLY) should be added to the step count. To our knowledge, the proposed sequence uses the minimum number of steps ever reported for an architecture that can retain input values. We performed repeated circuit simulations and analyzed the behavior of the circuit in the presence of variability and RTN, considering both the IMPLY and the SIMPLY implementations, the latter including the comparator and the control logic. We show that even if the energy improvement is smaller than that of the single operation alone, the SIMPLY implementation requires on average a third of the energy required by the IMPLY. The energy per sum operation is reported in Table 3 for every input combination. Compared to other works in the literature, of which the salient features are reported in Table 5, the IMPLY and SIMPLY results are coherent with experimental and physics-based simulations results. Other simulation works based on different architectures (e.g., MAGIC) and employing simplified general-purpose models, show lower energy estimates and shorter computation times than our results, probably due to the faster clock frequencies and lower current compliances considered in these works, besides the difference in architecture. Moreover, as remarked in Section II these estimates may not be very reliable when considering very fast voltage pulses. Currently, a hybrid FET-RRAM design (listed in Table 5) shows the best performance in terms of energy and computation speed. However, these estimates were obtained considering an extremely simplified RRAM model, and the proposed solution is not compatible with crossbar implementation which is an important requirement to enable the design of area efficient structures. In addition, the energy reduction provided by SIMPLY shifts the

TABLE 5. Comparison among the proposed and existing full adders in the literature.

Author(s)	Type of Logic (exp / sim)	Physics-Based Model	# Devices	Feasible in Crossbar	Energy* (estimated / reported)	# Elementary Steps	Delay* (estimated / reported)	Retains input values
Siemon et al. [23]	IMPLY (sim)	YES	8 RRAM	YES	202 pJ (estimated)	19	3.61µs (estimated)	NO
Lehtonen et al. [21]	IMPLY (sim)	NO	8 RRAM	YES	-	136	-	YES
Kvatinsky et al. [9]	IMPLY (sim)	NO	9 RRAM	YES	-	23	9.1 µs (estimated)	NO
Kvatinsky et al. [9]	IMPLY (sim)	NO	6 RRAM	YES	-	29	11.5 µs (estimated)	NO
Talati et al. [24]	MAGIC (sim)	NO	5 RRAM	YES	650 fJ (estimated)	13	19.5 ns (reported at 0.7 GHz)	NO
Talati et al. [24]	MAGIC (sim)	NO	10 RRAM	YES	800 fJ (estimated)	16	16.9 ns (reported at 0.7 GHz)	NO
Talati et al. [24]	MAGIC (sim)	NO	19 RRAM	YES	650 fJ (estimated)	13	20.8 ns (reported at 0.7 GHz)	NO
Talati et al. [24]	MAGIC (sim)	NO	10 RRAM	YES	500 fJ (estimated)	10	16.9 ns (reported at 0.7 GHz)	NO
Cheng et al. [12]	IMPLY (exp)	-	8 RRAM	YES	19.5 pJ (reported)	27	54 µs (reported)	NO
Puglisi et al. [10]	IMPLY (sim)	YES	9 RRAM	YES	6.4 nJ (reported)	43	345 ns (reported)	YES
This Work	**IMPLY (sim)**	**YES**	**8 RRAM**	**YES**	**518 pJ**	**28**	**560 ns**	**YES**
This Work	**SIMPLY (sim)**	**YES**	**8 RRAM**	**YES**	**172 pJ (0.172 fJ)****	**28**	**920 ns (92 ns)****	**YES**
Junsangsri et al. [25]	CMOS LiM (sim)	YES (FET) NO (RRAM)	41 FET + 4 RRAM	NO	2.2 fJ (reported – excludes RRAM energy)	-	52 ps (reported – excludes RRAM delay)	YES

* Reported means that the value was explicitly reported by the authors in the paper. Estimated means that the corresponding value has been inferred from data in the paper, although not explicitly reported by the authors. ** Projections considering I_C = 10nA and clock frequency = 0.5GHz [6, 28-29].

energy bottleneck from the IMPLY to the FALSE operation. Therefore, SIMPLY allows device/circuit requirements co-design with the aim of lowering the energy consumption also of the FALSE operation [27], which is the same for both for SIMPLY and the ordinary IMPLY, see Table 2. Thus, efforts can be directed at the device-level design to develop devices that achieve higher R_{HRS} by using lower $|V_{FALSE}|$, hence reducing the energy necessary for the FALSE operation. Further energy reductions can be achieved by lowering the current compliance [28] (increasing both R_{HRS} and R_{LRS}) and by using shorter pulses [6], [29]. We projected the energy per sum operation at lower I_C (10nA) and faster clock frequency (0.5 GHz) and showed that SIMPLY could potentially reach CMOS level performance, and even largely overcome them when the von Neumann bottleneck effects are considered.

VI. CONCLUSION

In this work we introduced a novel smart LIM scheme, SIMPLY, applicable to resistive switching memory technologies. We simulated its performance using a physics-based compact model calibrated using experimental data and showed that it greatly improves the reliability of the ordinary IMPLY logic, while also drastically lowering the energy consumption both when considering the single operation and the 1-bit full addition. In addition, the energy projections at lower I_C, show that SIMPLY is a viable solution for ultra-low power computing, ideal for IoT applications.

REFERENCES

[1] S. Li, C. Xu, Q. Zou, J. Zhao, Y. Lu, and Y. Xie, "Pinatubo: A processing-in-memory architecture for bulk bitwise operations in emerging non-volatile memories," in *Proc. 53rd ACM/EDAC/IEEE Design Autom. Conf. (DAC)*, 2016, pp. 1–6, doi: 10.1145/2897937.2898064.

[2] S. Kvatinsky, "Real processing-in-memory with memristive memory processing unit (mMPU)," in *Proc. IEEE 30th Int. Conf. Appl. Specific Syst. Archit. Process. (ASAP)*, 2019, pp. 142–148, doi: 10.1109/ASAP.2019.00-10.

[3] J. Backus, "Can programming be liberated from the Von Neumann style?: A functional style and its algebra of programs," *Commun. ACM*, vol. 21, no. 8, pp. 613–641, Aug. 1978, doi: 10.1145/359576.359579.

[4] H.-S. P. Wong *et al.*, "Metal–oxide RRAM," *Proc. IEEE*, vol. 100, no. 6, pp. 1951–1970, Jun. 2012, doi: 10.1109/JPROC.2012.2190369.

[5] 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, Apr. 2010.

[6] C. Wang *et al.*, "Ultrafast RESET analysis of HfO_x-based RRAM by sub-nanosecond pulses," *Adv. Electron. Mater.*, vol. 3, no. 12, 2017, Art. no. 1700263, doi: 10.1002/aelm.201700263.

[7] F. M. Puglisi, U. Celano, A. Padovani, W. Vandervorst, L. Larcher, and P. Pavan, "Scaling perspective and reliability of conductive filament formation in ultra-scaled HfO_2 resistive random access

memory," in *Proc. IEEE Int. Rel. Phys. Symp. (IRPS)*, 2017, pp. 1–5, doi: 10.1109/IRPS.2017.7936390.

[8] P.-Y. Chen, Z. Li, and S. Yu, "Design tradeoffs of vertical RRAM-based 3-D cross-point array," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 24, no. 12, pp. 3460–3467, Dec. 2016, doi: 10.1109/TVLSI.2016.2553123.

[9] 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, doi: 10.1109/TVLSI.2013.2282132.

[10] F. M. Puglisi, L. Pacchioni, N. Zagni, and P. Pavan, "Energy-efficient logic-in-memory I-bit full adder enabled by a physics-based RRAM compact model," in *Proc. 48th Eur. Solid-State Device Res. Conf. (ESSDERC)*, 2018, pp. 50–53, doi: 10.1109/ESSDERC.2018.8486886.

[11] A. Raghuvanshi and M. Perkowski, "Logic synthesis and a generalized notation for memristor-realized material implication gates," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, 2014, pp. 470–477, doi: 10.1109/ICCAD.2014.7001393.

[12] L. Cheng *et al.*, "Reprogrammable logic in memristive crossbar for in-memory computing," *J. Phys. D, Appl. Phys.*, vol. 50, no. 50, Dec. 2017, Art. no. 505102, doi: 10.1088/1361-6463/aa9646.

[13] T. Zanotti, F. M. Puglisi, and P. Pavan, "Circuit reliability of low-power RRAM-based logic-in-memory architectures," in *Proc. Int. Integr. Rel. Workshop (IIRW)*, 2019, pp. 1–5, doi: 10.1109/IIRW47491.2019.8989875.

[14] F. M. Puglisi, T. Zanotti, and P. Pavan, "SIMPLY: Design of a RRAM-based smart logic-in-memory architecture using RRAM compact model," in *Proc. 49th Eur. Solid-State Device Res. Conf. (ESSDERC)*, 2019, pp. 130–133, doi: 10.1109/ESSDERC.2019.8901731.

[15] 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, doi: 10.1109/TCSII.2015.2433536.

[16] F. M. Puglisi, N. Zagni, L. Larcher, and P. Pavan, "Random telegraph noise in resistive random access memories: Compact modeling and advanced circuit design," *IEEE Trans. Electron Devices*, vol. 65, no. 7, pp. 2964–2972, Jul. 2018, doi: 10.1109/TED.2018.2833208.

[17] F. M. Puglisi, T. Zanotti, and P. Pavan, "Unimore resistive random access memory (RRAM) Verilog-A model," *nanoHUB*, Jun. 2019, doi: 10.21981/15GF-KX29.

[18] S. Yu, B. Gao, Z. Fang, H. Yu, J. Kang, and H.-S. P. Wong, "A neuromorphic visual system using RRAM synaptic devices with sub-pJ energy and tolerance to variability: Experimental characterization and large-scale modeling," in *Proc. Int. Electron Devices Meeting*, 2012, pp. 1–4, doi: 10.1109/IEDM.2012.6479018.

[19] B. R. Russell and A. N. Whitehead, *Principia Mathematica*, vol. 1. Cambridge, U.K.: Cambridge Univ. Press, 1910.

[20] E. Lehtonen and M. Laiho, "Stateful implication logic with memristors," in *Proc. IEEE/ACM Int. Symp. Nanoscale Archit.*, 2009, pp. 33–36, doi: 10.1109/NANOARCH.2009.5226356.

[21] E. Lehtonen, J. H. Poikonen, and M. Laiho, "Two memristors suffice to compute all Boolean functions," *Electron. Lett.*, vol. 46, no. 3, pp. 239–240, Feb. 2010, doi: 10.1049/el.2010.3407.

[22] Y. Li, W. Chen, W. Lu, and R. Jha, "Impact of coupling capacitance on read operation of RRAM devices in 1D1R crossbar architectures," in *Proc. IEEE 57th Int. Midwest Symp. Circuits Syst. (MWSCAS)*, 2014, pp. 989–992, doi: 10.1109/MWSCAS.2014.6908583.

[23] A. Siemon *et al.*, "Stateful three-input logic with memristive switches," *Sci. Rep.*, vol. 9, no. 1, pp. 1–13, Oct. 2019, doi: 10.1038/s41598-019-51039-6.

[24] N. Talati, S. Gupta, P. Mane, and S. Kvatinsky, "Logic design within memristive memories using memristor-aided logic (MAGIC)," *IEEE Trans. Nanotechnol.*, vol. 15, no. 4, pp. 635–650, Jul. 2016, doi: 10.1109/TNANO.2016.2570248.

[25] P. Junsangsri, J. Han, and F. Lombardi, "Logic-in-memory with a nonvolatile programmable metallization cell," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 24, no. 2, pp. 521–529, Feb. 2016, doi: 10.1109/TVLSI.2015.2411258.

[26] J. E. Stine *et al.*, "FreePDK: An open-source variation-aware design kit," in *Proc. IEEE Int. Conf. Microelectron. Syst. Edu. (MSE)*, 2007, pp. 173–174, doi: 10.1109/MSE.2007.44.

[27] T. Zanotti, F. M. Puglisi, and P. Pavan, "Smart logic-in-memory architecture for ultra-low power large fan-in operations," in *Proc. IEEE Int. Conf. Artif. Intell. Circuits Syst. (AICAS)*, Genova, Italy, 2020.

[28] J. Zhou, F. Cai, Q. Wang, B. Chen, S. Gaba, and W. D. Lu, "Very low-programming-current RRAM with self-rectifying characteristics," *IEEE Electron Device Lett.*, vol. 37, no. 4, pp. 404–407, Apr. 2016, doi: 10.1109/LED.2016.2530942.

[29] B. J. Choi *et al.*, "High-speed and low-energy nitride memristors," *Adv. Funct. Mater.*, vol. 26, no. 29, pp. 5290–5296, 2016, doi: 10.1002/adfm.201600680.