A Semi-Serial Topology for Compact and Fast IMPLY-based Memristive Full Adders

N. TaheriNejad*, T. Delaroche†‡, D. Radakovits *, and S. Mirabbasi‡
* TU Wien, Vienna, Austria
E-mail: {nima.taherinejad, david.radakovits}@tuwien.ac.at
† University of Bordeaux INP, Bordeaux, France
E-mail: theophile.delaroche@gmail.com
‡ University of British Columbia, Vancouver, Canada
E-mail: shahriar@ece.ubc.ca

***Abstract*—Memristive systems are among the emerging technologies that hold a great promise. They are compact, CMOS compatible, easy to fabricate and can serve for storage as well as computation purposes. Adders are one of the most basic and critical building blocks of any computing system. One of the main application areas of memristors is in Material Implication (IMPLY) based logic. IMPLY-based adders are implemented either in serial, which has a compact implementation but needs many steps for calculation, or in parallel, which is fast, however, requires a large number of memristors. In this paper we propose an IMPLY-based adder topology and its respective addition algorithm which is 54-to-65% faster than serial adders and requires 46-to-76% less memristors than parallel adders. This topology is a favorable candidate for applications where neither speed, nor cost (i.e., area or number of memristors) could be compromised to gain the required performance.**

I. Introduction

Memristors, given their characteristics, are a promising base for new, fast, compact and efficient computing systems. While memristors are nominally used in memory applications [1, 2, 3, 4, 5], they can also be employed in logic, which makes them an ideal candidate for In-Memory Processing (IMP) systems, performing memory and logic operations on the same hardware without the need of data transport to a Central Processing Unit (CPU). The logic family that is used in the proposed design is Material Implication (IMPLY) [6], which is one of the most prominent logic families used in memristive IMP systems. Since in IMPLY the logic value is stored in the resistance, i.e., state of the memristor, rather than in voltage or current, it is considered as a so called *stateful* logic [7]. As its name suggests, IMPLY performs a material implication $b = a \rightarrow b$, where a and b are both input memristors before the logic operation and the result of the implication is stored in b, making it the output memristor after the logic operation. In IMPLY, the output (b') is false ('0') only if $a = 1$ and $b = 0$, in which logic '1' is mapped to R_{on} and logic '0' is mapped to R_{off}. For all other values of a and b the output (b') is valid ('1'). For more information regarding the details of the operation in IMPLY logic, we refer the readers to [6, 7, 8]. Because of its structure and operation principles, IMPLY is highly suitable for standard crossbar design, a structure broadly used in memory design.

Adders are among the most critical building blocks of all computational systems, and they are used in practically every processed command. Therefore, making adders more compact, more efficient, and faster is an important step towards improvement of the memristive IMP systems. However, the improvement space in IMPLY-based adders seems saturated as the progress has been considerably scarce and slow in recent years. Therefore, we contribute in advancing the state-of-the-art by introducing a new topology and a new algorithm for it. The proposed adder is faster than existing serial adders and requires less memristors compared to its parallel counterparts. Moreover, it has the best Figure of Merit (FoM) among all existing IMPLY-based adders.

Among existing adder architectures, serial topology shown in Figure 1(a) is the most popular approach [7, 9, 8, 10, 11]. The main reason for its widespread use is the fact that it has the most compatibility with cross-bar, that is, minimum changes or peripheral elements are needed to use it in a cross-bar. The main advantage of this topology is its low number of memristors. In [11], a serial adder with the smallest number of memristors to date, that is, $2n + 3$ memristors for an n-bit adder, is proposed. This is achieved by re-using input memristors to store the output value too. Number of steps required in this design for an addition is $22n$.

Parallel topology is the other prominent approach, shown in Figure 1(b), which was first proposed in [8]. The main feature of this topology is its reduced calculation time, $9n$, which comes at the cost of increased number of memristors needed, $5n + 18$. An alternative parallel adder topology is recently proposed in [12], where the total number of memristors in this design is $4n + 1$ and they manage to perform an addition in $5n + 18$ steps[1].

II. Proposed Full Adder Structure

A. Topology

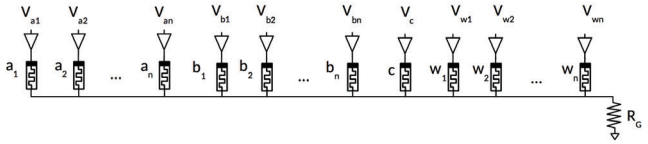
In the serial structures reviewed in Section I, the main goal of the designers has been to minimize the number of work

[1]It should be noted that in [12] the number of required steps is quoted as $5n + 16$, however, we believe that this is excluding the 2 steps required for initialization of the work memristors.

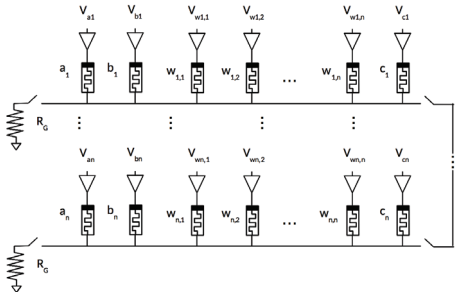
TABLE I

EXECUTIONAL STEPS OF ADDITION IN THE PROPOSED SEMI-PARALLEL TOPOLOGY. VALID: $10n+2$ STEPS, $2n+6$ MEMRISTORS AND $c=\overline{c_{in}}$. WORK MEMRISTORS AND c CAN CONNECT TO ANY OF THE TWO SECTIONS. THE OPERATION IN BLUE (UNNUMBERED STEPS) ARE PERFORMED ONLY FOR THE VERY FIRST AND VERY LAST BIT OF THE OPERATION.

Steps	Operation Executed in:		Equivalent Logic	
	Section 1	Section 2	Section 1	Section 2
1	$c=0$ & $w_1=w_2=0$	$w_3=w_4=0$	False(c) & False(w_1,w_2)	False(w_3,w_4)
-	$c_{in}\rightarrow c$		$c=\overline{c_{in}}$	
2	$a\rightarrow w_1=w_1'$	$b\rightarrow w_3=w_3'$	$w_1'=\overline{a}$	$w_3'=\overline{b}$
3	$a\rightarrow w_3'=w_3''$	$w_1'\rightarrow b=b'$	$w_3''=a\rightarrow\overline{b}$	$b'=\overline{a}\rightarrow b$
4	$c\rightarrow w_2=w_2'$	$w_3''\rightarrow w_4=w_4'$	$w_2'=\overline{\overline{c}}=c$	$w_4'=\overline{a\rightarrow\overline{b}}$
5	$a=w_1=0$	$b'\rightarrow w_4'=w_4''$	False(a,w_1)	$w_4''=(\overline{a}\rightarrow b)\rightarrow\overline{(a\rightarrow\overline{b})}=\overline{a\oplus b}$
6	$w_3''\rightarrow w_2'=w_2''$	$w_4''\rightarrow c=c'$	$w_2''=\overline{(a\rightarrow\overline{b})}\rightarrow c$	$c'=\overline{a\oplus b}\rightarrow\overline{c}$
7	$c'\rightarrow a=a'$	$w_2''\rightarrow w_1=w_1'$	$a'=\overline{\overline{a\oplus b}\rightarrow\overline{c}}$	$w_1'=\overline{(a\rightarrow\overline{b})\rightarrow c}$
8	$c_{in}=0$ & $c=w_3=0$	$b'\rightarrow w_2''=w_2'''$	False(c_{in}) & False(c,w_3)	$w_2'''=\overline{(\overline{a}\rightarrow b)\rightarrow[(a\rightarrow\overline{b})\rightarrow c]}$
9	$w_1'\rightarrow w_3=w_3'$	$b'\rightarrow c=c'$	$w_3'=(a\rightarrow\overline{b})\rightarrow c$	$c'=\overline{\overline{a}\rightarrow b}=\overline{\overline{b}\rightarrow a}$
10	$w_2'''\rightarrow a'=a''$	$w_3'\rightarrow c'=c''$	$a''=\big((\overline{a}\rightarrow b)\rightarrow[(a\rightarrow\overline{b})\rightarrow c]\big)\rightarrow\overline{\overline{a\oplus b}\rightarrow\overline{c}}=S(Sum)$	$c''=[(a\rightarrow\overline{b})\rightarrow c]\rightarrow\overline{(\overline{b}\rightarrow a)}=\overline{C_{out}}$
-	$c\rightarrow c_{in}$		$c_{in}=\overline{c}=C_{out}$	



(a) Serial



(b) Parallel

Fig. 1. Typical IMPLY-based Full Adder topologies.

memristors. This comes at the cost of speed. Furthermore, the fully serial approach leads to having a large number of steps, i.e., longer calculation time. As mentioned in Section I, another approach is the parallel one which comes with a large number of memristors and switches. In this work, we propose the topology shown in Figure 2. This topology is similar to the

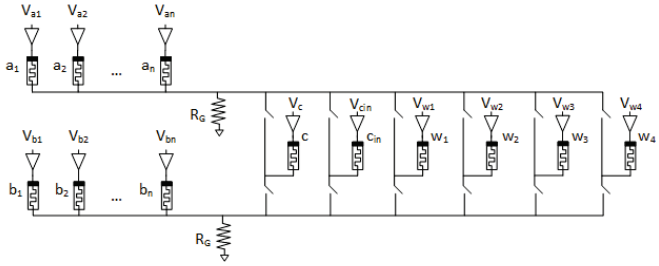


Fig. 2. The topology of the proposed semi-serial full-adder.

serial one, however, the input variables, a_i and b_i, constitute two separate sections. There is a third section as well, where carry and work memristors sit. This third section can be selectively connected to each of the two input variable sections (or to none). This third section uses five work memristors (W_1 to W_4 and c) for the entire adder. Thus the total number of memristors is $n+n+1+5=2n+6$. This number is significantly less than parallel approaches where the number of work memristors is proportional to the size of the adder, n. Most serial adders use only two work memristors, however, our three additional work memristors results in a negligible overhead since in practical applications $n=32$ or $n=64$ this would mean less than 3% additional area. Lastly, similar to [11] and [12], we reuse variables a (n-bit variable) and c_{in} to store the output sum and carry. To calculate the Sum (S) and Carry-out (C_{out}) we use the following presentations;

$$S=\Big[(\overline{a}\rightarrow b)\rightarrow\Big((a\rightarrow\overline{b})\rightarrow c\Big)\Big]\rightarrow\Big(\overline{\overline{(a\oplus b)}\rightarrow\overline{c}}\Big) \quad (1)$$

$$C_{out}=\Big[\overline{\Big((a\rightarrow\overline{b})\rightarrow c\Big)\rightarrow\overline{(\overline{b}\rightarrow a)}}\Big] \quad (2)$$

The detailed steps of the algorithm for the proposed topology are presented in Table I. In this algorithm, it is assumed $\overline{c}$ is provided in the input and is propagated to the next bit. Hence, on top of the 10 steps for each bit ($10n$) we consider one additional step for the initial inversion at the first bit (that is, c_{in} to $c=\overline{c_{in}}$) and one for the the final inversion at the last bit (that is, c to $c_{out}=\overline{c}$). These steps, which are taken only for the first bit and the last bit calculation, respectively, are shown in Table I using blue color. Therefore, the overall number of steps in this design is $10n+2$.

In this topology each bit is calculated one after another (in a serial fashion) and has some structural similarities with the serial topology (in terms of connections and number of memristors), however, the structure of the work memristors section is different than that of serial topology. Hence, we refer to the proposed structure as semi-serial topology. We note that the work memristor section requires a 1T1M crossbar architecture (and external Complementary Metal-Oxide Semiconductor (CMOS) switches). Even though the operands could

TABLE II
SETUP VALUES FOR VTEAM MODEL.

Parameter	v_{off}	v_{on}	α_{off}	α_{on}	R_{off}	R_{on}
Value	0.7 V	-10 mV	3	3	1 MΩ	10 kΩ
k_{on}	k_{off}	w_{off}	w_{on}	w_C	a_{off}	a_{on}
-0.5 nm/s	1 cm/s	0 nm	3 nm	107 pm	3 nm	0 nm

be implemented in a 1M crossbar architecture too, it is simpler to implement the entire design using a 1T1M crossbar.

B. Simulations

1) Setup: Using the LTSpice software and the VTEAM model [13] imported to SPICE [14], the algorithm is tested with the parameters in Table II. The parameters in the table are selected such that they describe the behavior of real memristors we have at hand[2]. Given the memristor model setup, to ensure proper IMPLY logic parameters, the following values are used in our simulations: $\{V_{SET}, V_{COND}, V_{RESET}, R_G,\ t_{pulse}\}$ = $\{1V,\ 900mV$,-$5V,\ 40k\Omega,\ 30\mu s\}$.

2) Results: The proposed algorithm is first tested as a single-bit adder with all of the different input combinations, which resulted in the correct outputs. Figure 3 shows the simulation for the single-bit adder with the inputs $a = 0,\ b = 0,\ c_{in} = 1 = \bar{c}$. As expected, the results of the computation are: Sum = $a = 1$ and $C_{out} = \bar{c} = 0$. By simulating all of the different input combinations for the proposed one bit full adder, we calculated the average energy consumption to be 9.87 nJ per bit, excluding the one time inversion of carry bit at the very first bit input and the very last bit output. The overhead of these inversions is 1.33 nJ leading to the overall energy consumption of $9.87n+1.33$ nJ for the n-bit full adder. Next, we simulated a 4-bit full adder to validate our design. The simulation example shown in Figure 4 gives the expected outputs, Sum = $a_{4-1} = 0110$ and $C_{out} = \bar{c} = 1$, for its inputs; $a_{4-1} = 1001,\ b_{4-1} = 1100,\ c = 0 = \overline{c_{in}}$.

C. Scope and Limitations

We note that this work mainly concerns topology and algorithm design for IMPLY-based adders and regards the basic IMPLY gates at a behavioral level. That is, the details of IMPLY operations in practice are outside the scope of this work. Thus, practical aspects such as parasitic elements, variations in memristance, and noise are not studied.

Moreover, we acknowledge that a fully fair and comprehensive comparison requires implementation or post-layout simulations, which is not possible to us, nor is reported by others. Therefore, currently some factors such as area or processing time can be only estimated or compared by proxy[3]. For example, due to additional switches, the proposed adder is more complex compared to a traditional serial adder. However, these switches could be implemented underneath the memristors array and thus not affect the die area. With respect to the parallel adder, the complexity depends on the number of bits. In a parallel adder, with the increase in the number of bits the number of required switches also increase linearly ($2n$ for an n-bit adder) whereas in the proposed design the number of switches is constant (12 switches for any n). Therefore, for any $n > 6$, the proposed adder is expected to be less complex.

[2]Our memristors are tungsten chalcogenide Resistive Random Access Memorys (ReRAMs) produced by KNOWM [15]. The data-sheet of the used memristors including more detailed information can be found at [16].

[3]Area using the number of memristors and processing time via number of steps.

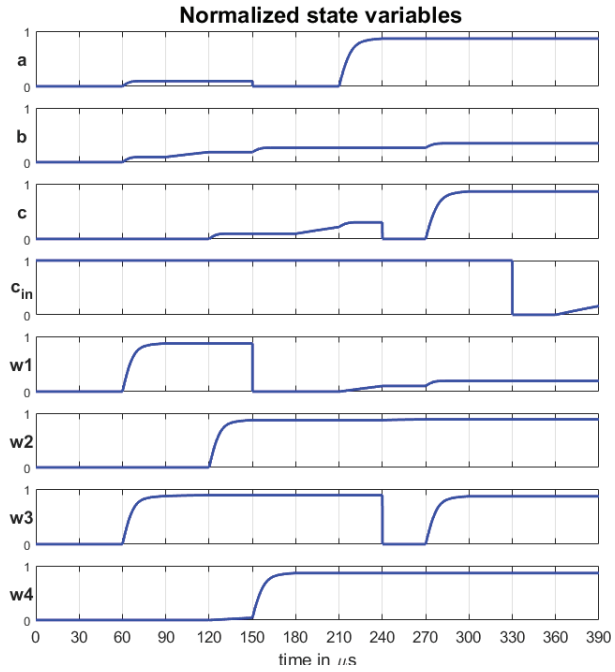


Fig. 3. One bit adder simulation with $a = 0,\ b = 0,\ c_{in} = 1$, each step is $30\mu s$ long in the simulation settings.

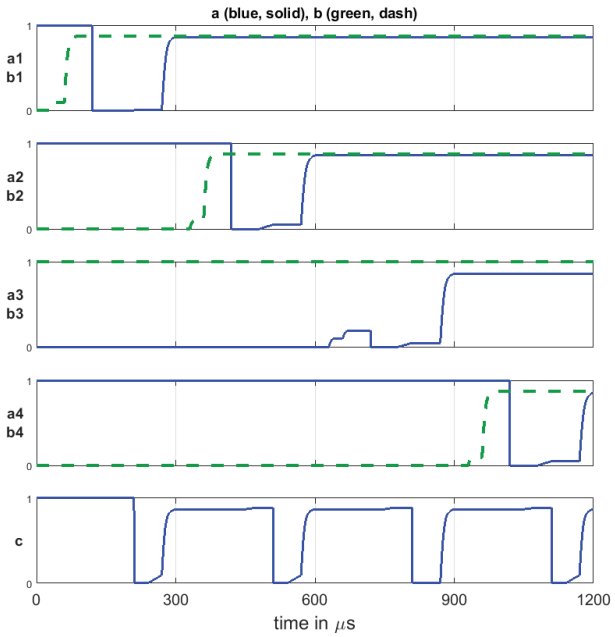


Fig. 4. 4-bit adder simulation with $a_{4-1} = 1011,\ b_{4-1} = 0100, c = 0 = \overline{c_{in}}$, the calculation time for one bit is $300\mu s$.

TABLE III
SUMMARY OF COMPARISONS BETWEEN THE PROPOSED ALGORITHM AND OTHER IMPLY-BASED ADDERS.

Designs	Number of Memristors					Number of Steps			FoM	
	Reused	Work	Total	$n = 32$	Imp.	Total	$n = 32$	Imp.		Imp.
Serial [8]	1	2	$3n + 3$	99	29%	$29n$	928	65%	91.9	76%
Serial [10]	1	2	$3n + 3$	99	29%	$23n$	736	56%	72.9	69%
Serial [11]	$n + 1$	2	$2n + 3$	67	-4%	$22n$	704	54%	47.2	52%
Serial [12]	$n + 1$	2	$2n + 3$	67	-4%	$23n$	736	56%	49.3	54%
Parallel [8]	1	$6n$	$9n$	288	76%	$5n + 18$	178	-45%	51.3	56%
Parallel [12]	$n + 1$	$2n$	$4n + 1$	129	46%	$5n + 18$	178	-45%	23.0	2%
Proposed	$n + 1$	5	$2n + 6$	70	-	$10n + 2$	322	-	22.5	-

III. COMPARISON

To explore the advantages and disadvantages of the proposed design, we have compared it with the most recent IMPLY-based full adders. A summary of this comparison is presented in Table III. In these comparisons, FoM using the product of the number of memristors and the number of steps, divided by a thousand. This facilitates comparison in system where are and speed are equally important. The percentages of improvement are calculated based on $\frac{P_{worse}-P_{better}}{P_{worse}} \times 100$, where P_{better} is the respective parameter of the better design and P_{worse} is that of the worse design used as the base for comparison. We note that we do not compare the power consumption due to two reasons. First, many of the works do not report any values regarding the power consumption. Second, power consumption depends on the type of the memristors used and their characteristics, and the comparisons between adders simulated with different models is not fair and does not represent their merit.

As shown in Table III, in terms of FoM, the proposed design is better than all other designs. In terms of improvements in the number of memristors compared to parallel designs, the proposed approach is 50% and 78% better than [12] and [8] respectively. In the case of other serial designs, when better, the proposed architecture is 29% smaller. Note that for the serial designs in [11] and [12] have 3 memristors less than the proposed design. However, this difference is quite negligible (only 4%) and in adders with large number of bits it is zero (for n approaching infinity). In terms of number of steps, the proposed design is 54 to 65% better than other serial designs. Compared to parallel designs it is 45% slower, however, we note that the proposed topology has a more compact area, i.e., less number of memristors. In both cases, the FoM has improved. Hence, for applications where both area and speed have equal weights, or where the area of the design has a larger weight, the proposed design would compare favorably with all other designs.

IV. CONCLUSION

In this paper, we propose an IMPLY-based adder topology, along with its associated algorithm to run the addition operation. The proposed semi-serial adder uses $2n + 6$ memristors and the addition of two n-bit inputs is completed in $10n + 2$ steps. This gives the proposed solution an edge — in terms of FoM — compared to all IMPLY-based adders. Compared to serial designs, the improvement are anywhere between 29 to 65%. Compared to parallel adders, it is 45% slower, however, this compromise in speed is compensated by a larger improvements in the number of required memristors.

REFERENCES

[1] H. Kim et al. Memristor-based multilevel memory. In *CNNA2012*, pages 1–6, Feb 2010.

[2] M. Zangeneh and A. Joshi. Design and optimization of nonvolatile multibit 1T1R resistive RAM. *VLSI2014*, 22(8):1815–1828, Aug 2014.

[3] N. Taherinejad et al. Memristors' potential for multi-bit storage and pattern learning. In *EMS*, pages 450–455, 2015.

[4] N. Taherinejad et al. Fully digital write-in scheme for multi-bit memristive storage. In *CCEC*, pages 1–6, 2016.

[5] Kuk-Hwan Kim et al. A functional hybrid memristor crossbar-array/cmos system for data storage and neuromorphic applications. *Nano letters*, 12(1):389–395, 2011.

[6] Julien Borghetti et al. memristiveswitches enable statefullogic operations via material implication. *Nature*, 464(7290):873–876, 2010.

[7] Eero Lehtonen and Mika Laiho. Stateful implication logic with memristors. In *NANOARCH2009*, pages 33–36. IEEE Computer Society, 2009.

[8] Shahar Kvatinsky et al. Memristor-based material implication (imply) logic: design principles and methodologies. *VLSI2014*, 22(10):2054–2066, 2014.

[9] Bickerstaff K'A and Earl E Swartzlander. Memristor-based arithmetic. In *ACSSC2010*, 2010.

[10] Mehri Teimoory et al. Optimized implementation of memristor-based full adder by material implication logic. In *ICECS2014*, pages 562–565, 2014.

[11] S. G. Rohani and N. TaheriNejad. An improved algorithm for imply logic based memristive full-adder. In *CCECE2017*, pages 1–4, April 2017.

[12] Ahmad Karimi and Abdalhossein Rezai. Novel design for a memristor-based full adder using a new imply logic approach. *Journal of Computational Electronics*, 17(3):1303–1314, Sep 2018.

[13] Shahar Kvatinsky et al. VTEAM: A general model for voltage-controlled memristors. *CSII2015*, 62(8):786–790, 2015.

[14] https://www.ict.tuwien.ac.at/staff/taherinejad/projects/memristor/files/vteam.sub, March 2018.

[15] https://knowm.org/, February 2019.

[16] https://knowm.org/wp-content/uploads/dm8-16dip-bs-afw.pdf, February 2019.