An IMPLY-based Semi-Serial Approximate In-Memristor Adder

Fabian Seiler* Nima TaheriNejad+*
*Technische Universität Wien (TU Wien), Austria, +Heidelberg University, Germany
fabian.seiler@student.tuwien.ac.at, nima.taherinejad@ziti.uni-heidelberg.de

***Abstract*—To alleviate the Von Neumann bottleneck, new technologies and computing paradigms have been a hot topic in research and development in recent years. Memristors offer new innovative possibilities from technological and computational points of view. They can store data well and are suitable for in In-Memory Computation (IMC) since they are able to perform logical operations in memory. Another emerging computing paradigm to reduce computing time and area consumption is approximate computing, which is used in error-resistant applications. Here, we propose a novel approximated full adder hat uses the stateful logic Material Implication (IMPLY) in a semi-serial structure. We embedd this full adder in a Ripple Carry Adder (RCA) that we then evaluate on the circuit-level. The error metrics were evaluated and compared to State-of-the-Art (SoA) IMPLY-based adders. At 8-bit our approach requires up to 29% fewer steps and up to 34% less energy compared to the exact algorithm, while the Normalized Median Error Distance (NMED) is less than 0.01 for most scenarios. The proposed adder is applied in image processing and the respective quality metrics are calculated. All of the tested approximation degrees create a satisfactory result since the Peak Signal-to-Noise Ratio (PSNR) is over 30 dB. Thanks to the proposed approach, we save more than 13.5mJ of energy in gray-scale filtering of a 684×912 8-bit image compared to the exact calculations.**

I. Introduction

There is currently a stagnation in the improvement of computer performance caused by problems such as the slowdown of Moore's Law [1] or the von Neumann bottleneck. As a consequence of it new emerging technologies and paradigms, like the memristor technology, in-memory computing, and approximate computation, arose. For approximated computation, an inaccuracy in the calculation is accepted in order to improve speed, area, and energy consumption [1]–[3]. Approximated computation is used in error-resistant applications such as image and video processing. Other application areas such as machine learning, pattern recognition, communication, data mining and robotics are also relevant [1], [4]. With In-Memory Computation (IMC) it is possible to bypass the von Neumann's bottleneck since the computations can be performed directly in memory. The memristor is ideally suited for IMC, since it can store data through its resistive state [5]–[7] and logical operations can be performed with it [8], [9]. Among various memristive logics, the stateful logic Material Implication (IMPLY) is compatible with the crossbar array and is ideal for IMC [8], [10].

979-8-3503-3757-0/23/$31.00 ©2023 IEEE

In this work, we present a novel approximated addition algorithm that is based on the semi-serial topology [11]. With this approach, we were able to drastically reduce energy consumption and the number of computational steps required in comparison to the exact algorithm.

In Section II, we review the necessary background and the current State-of-the-Art (SoA). A detailed explanation of the algorithm and the design methodology can be found in Section III. In Section IV, we simulated the presented algorithm on circuit-level, verified its functionality and performed an error analysis using standard metrics. A comparison with SoA full adders is done in Section V. We simulated three image processing applications and did a respective quality check with the presented full adders in Section VI. In Section VII, we conclude the paper and discuss future work.

II. Background

A. Memristor and IMPLY

A memristor is a two-terminal nonvolatile memory that stores its logical values as electrical resistance. The minimum (R_{on}) and maximum (R_{off}) resistance values of the memristor can be reached by the applied voltage and the direction of current flow [12]. One of the conventions is to assume the minimum resistance value is equivalent to a logical '1' and the maximum resistance value equal to a logical '0' [13]–[15]. Advantages of the memristor include low power consumption, as well as low write-time and small dimension of the device [16]–[18]. There are plenty of ways to perform calculations with memristors [19]–[21]. Among various logics [17], [22]–[24], in this work, we only use IMPLY, which established itself as the first stateful logic and was proposed by Hewlett Packard (HP) [8], [25], [26]. The basic structure to perform IMPLY operations is shown in Figure 1, where the resistive states of the input memristors a and b represent the logical inputs. The IMPLY operation is represented by $a \rightarrow b$ which corresponds to the truth table in Table I and stores the result in the b-memristor, overwriting the current state. For more information regarding the details of IMPLY logic, we refer the reader to [8], [16], [25], [26].

Full adders that are based on IMPLY can be divided into serial [8], [9], [13], parallel [13], [16], [25] and hybrid structures that combine serial and parallel [14], [27]. The serial structure consists of only one row (or column), which simplifies the design at the cost of its calculation speed. The parallel structure can compute different steps in parallel rows,

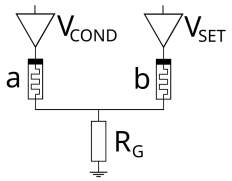

Fig. 1: Structure of an IMPLY gate [8]

TABLE I: Truth Table of IMPLY logic [8]

a	b	$b' = a \rightarrow b$
0	0	1
0	1	1
1	0	0
1	1	1

which can be connected through switches. This methodology enables rapid computations at the expense of increased spatial and energy requirements. Hybrid structures try to combine the advantages of parallel and serial topologies in an efficient manner. The semi-serial structure that was utilized in this work is a hybrid structure that achieves a well-balanced trade-off between energy consumption, area usage, and speed [11], [14]. This topology is displayed in Figure 2 and consists of two parallel rows with input memristors, which both can connect to four work memristors, c_{in} and c-memristor via switches. With the two separate sections, a parallelization of operations is possible. This structure only requires $2n + 6$ memristors and 12 Complementary Metal-Oxide Semiconductor (CMOS)-switches while it is able to compute an n-bit addition in only $10n + 2$ steps.

B. Approximate Computing and Quality Metrics

With approximate computation, we improve factors such as energy consumption, area usage, and speed but decrease accuracy as a trade-off. To evaluate the degree of inaccuracy, error metrics were used in SoA publications like [28]–[34]. The most important and used metrics in this work are Error Distance (ED), Error Rate (ER), Mean Error Distance (MED), Normalized Median Error Distance (NMED), and Median Relative Error Distance (MRED). There exist many variants of approximated full adders in different technologies [3], [31], [32], [35], [36]. Approximated adders based on memristors have been proposed in [37], [38] where they used the Memristor Ratioed Logic (MRL) [24]. Approximated full adders that utilized IMPLY have also been presented only very recently in [15], [34], where new approximate algorithms for the serial structure were proposed. Here, we propose a novel new algorithm for an approximated full adder in the semi-serial structure and compare our results with the SoA.

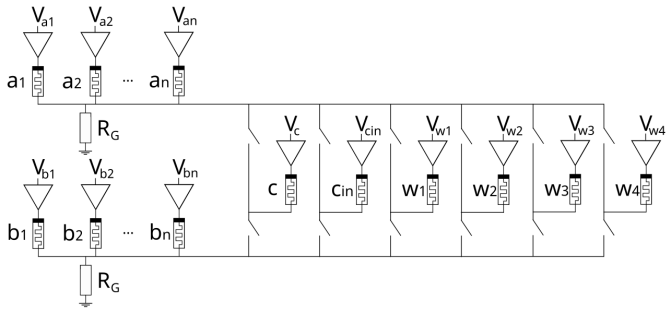

Fig. 2: IMPLY based semi-serial n-bit adder structure [14]

III. Proposed approximate full adder

Since this work is based on IMPLY logic, only IMPLY and FALSE operations are basic operations used. These two operations form a complete logic set, and we are able to emulate any Boolean logic with them [13], [39]. As we are working in the semi-serial structure from [11], we leveraged its inherent parallelization capability. We developed the approximation in this work by introducing an error in the truth table of C_{out} in the case [a,b,c] = "100". We additionally utilized the similarity $Sum \approx \overline{C_{out}}$, to only require one additional step to calculate Sum. With this approximation, we are creating three erroneous cases in the truth table of Sum. This leads to C_{out} having an ER of $\frac{1}{8}$ and Sum having an ER of $\frac{3}{8}$. The full truth table can be found in Table III. We determined the conjunctive normal form and found an efficient emulation of the boolean functions with IMPLY operations. The logical functions that we used are

$$Cout = bc + a = \overline{a} \rightarrow \overline{(b \rightarrow \overline{c})} \tag{1}$$

$$Sum = \overline{bc + a} = \overline{\overline{a} \rightarrow \overline{(b \rightarrow \overline{c})}} \tag{2}$$

As the Boolean operations NOT, OR and NAND can be emulated very efficiently, we used only those. As both OR and NAND require one negated input it is possible to combine them to save an additional step. To be able to read out the result of the calculation, we used the a-memristor to save Sum. So that the algorithm can be used in a Ripple Carry Adder (RCA) configuration, the algorithm was designed so that the C_{out} is stored in the c-memristor. The complete procedure of the proposed algorithm is listed in Table II. The step colored in blue is only executed once before the first iteration and resets the work memristors. This reset can be parallelized in repetitions of the algorithm. In the following steps, we are working in two sections in parallel to fully utilize the parallelization potential of the semi-serial structure. We first saved the inversions of the first input bit and the carry-in in the work memristors. We did that to emulate b NAND c and additionally free the c-memristor in the second step. In steps 3 and 4, we calculate C_{out} and store it in the c-memristor. Additionally, we are resetting the a-memristor. In the last step, we store the result of Sum in the a-memristor and reset the work memristors for the next iteration. This approximated algorithm requires $5n + 1$ steps and $2n + 3$ memristors for an n-bit calculation.

IV. Circuit-Level Simulations and Error Metrics

A. Simulation setup

We simulated the proposed approximated full adder in LTSPICE to confirm and verify the functionality of the algorithm.

TABLE II: Exact Procedure of the proposed approximated full adder algorithm, with $w1$ and $w2$ as work memristors

Steps	Section 1	Section 2	Equivalent Logic
-		$w_1 = w_2 = 0$	$False(w_1, w_2)$
1	$w_2^{'} = a \rightarrow w_2$	$w_1^{'} = c \rightarrow w_1$	$w_2 = \overline{a}$, $w_1 = \overline{c}$
2	$c = 0$	$w_1^{''} = b \rightarrow w_1^{'}$	$False(c)$, $w_1 = b \rightarrow \overline{c}$
3	$c^{'} = w_1^{''} \rightarrow c$		$c = \overline{b \rightarrow \overline{c}}$
4	$a = 0$	$c^{''} = w_2^{'} \rightarrow c^{'}$	$False(a)$, $c = \overline{a} \rightarrow (\overline{b \rightarrow \overline{c}}) = Cout$
5	$a^{'} = c^{''} \rightarrow a$	$w_1 = w_2 = 0$	$a = \overline{\overline{a} \rightarrow (\overline{b \rightarrow \overline{c}})} = Sum$, $False(w_1, w_2)$

TABLE III: The truth table of the exact and the proposed approximated full adder

a	b	c	Exact Sum	Exact Cout	Ax Sum	Ax Cout
0	0	0	0	0	1	0
0	0	1	1	0	1	0
0	1	0	1	0	1	0
0	1	1	0	1	0	1
1	0	0	1	0	0	1
1	0	1	0	1	0	1
1	1	0	0	1	0	1
1	1	1	1	1	0	1

TABLE IV: VTEAM setup parameter

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

TABLE V: IMPLY logic parameter

Parameter	V_{SET}	V_{RESET}	V_{COND}	R_G	t_{pulse}
Value	1 V	-1 V	900 mV	40 kΩ	30 μs

We used the Voltage-controlled ThrEshold Adaptive Memristor (VTEAM) model [40] implemented in SPICE [11], [41]. The model parameters we selected are listed in Table IV. We note that these parameters are the result of fitting the model to discrete memristors. This leads to slower operations and larger power consumption compared to integrated devices, similar to how discrete CMOS devices compare to their integrated counterparts. The specific parameters of the IMPLY logic that we used in this simulation are listed in Table V. The parameters were chosen this way because the same setup was already used in [13], [16], [34], allowing for a good comparison to existing approximated and exact full adder.

B. Simulation results

We simulated the algorithm for the approximated full adder that we presented in Section III with the parameters described above in the semi-serial topology from [11] on circuit-level. We tested all 8 possible input combinations of "AinBinCin" for functionality. A correct function is given if Sum and C_{out} at the end of the respective algorithm results in the same solution as described in the corresponding truth table. Each step of the algorithm takes $30\mu s$. C_{out} is calculated and stored in the c-memristor in the fourth step, which corresponds to the period between $120\mu s - 150\mu s$. We used the fifth step at $150\mu s - 180\mu s$ to calculate Sum and store it in the a-memristor. All input combinations produced the expected output. The output waveform of each memristor of the presented

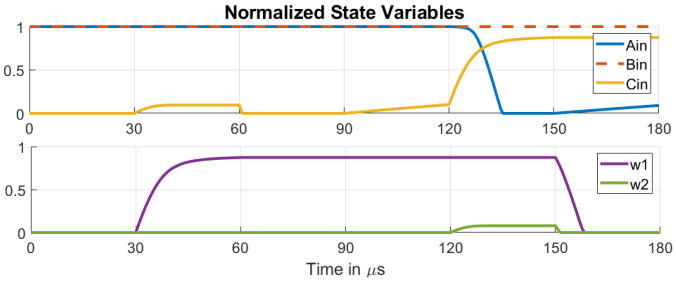

Fig. 3: Simulation of the proposed algorithm with "AinBinCin"="110"

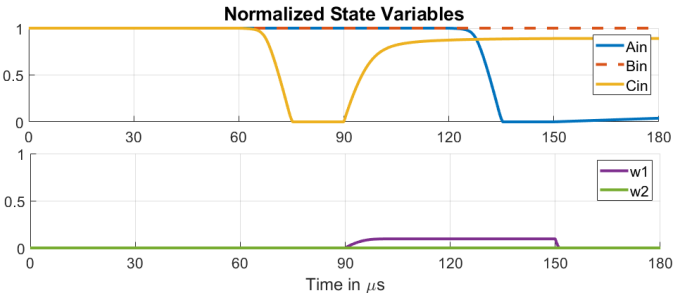

Fig. 4: Simulation of the proposed algorithm with "AinBinCin"="111", showing an approximated (erroneous by design) output

algorithm was plotted at cases "AinBinCin"="110" and "111" to show a correct calculation of Sum and C_{out} in the first case (Figure 3) and a calculation showing an error due to our chosen approximation (Figure 4). To ensure correct functionality of the full adder on circuit-level with multiple bits, we tested the algorithm in a 4-bit RCA. For this, we let the lowest two bits use the proposed algorithm and the higher two bits use the exact full adder algorithm for a semi-serial topology from [11]. Five random pairs of 4-bit numbers were added by our LT-SPICE simulation and the results agree with our theoretical calculations.

We calculated the energy consumption of the proposed algorithm with the LT-SPICE energy consumption tool. We simulated all possible input combinations a full adder could take on. The result is defined as the average value of all simulations. Since the first step of the algorithms is performed only once before the first execution, it is not considered in the energy consumption that is required per bit, because it is negligible with respect to several bits of computation. The first step consumes $55.5pJ$. The complete energy consumption of a RCA that includes the proposed algorithm is

$$Energy(n,k) = 1.6678k + 3.8435(n-k) + 0.865, \quad (3)$$

where k represents the number of approximated adders and n the total number of bits.

C. Error analysis

To evaluate the erroneous behavior of the approximated full adder we presented in this work, we use the error metrics

TABLE VI: Error Metrics of the algorithm in a 8/16/32-bit RCA with different approximation degrees

Approximated Algorithm	MED	NMED	MRED
8-bit RCA			
Ax FA: 1/8	0.5	0.0010	0.0027
Ax FA: 2/8	1.1250	0.0022	0.0062
Ax FA: 3/8	2.2500	0.0044	0.0125
Ax FA: 4/8	4.4688	0.0087	0.0252
Ax FA: 5/8	8.9121	0.0174	0.0514
Ax FA: 8/8	203.38	0.9159	0.3980
16-bit RCA			
Ax FA: 2/16	1.1469	0.0000087	0.0000024
Ax FA: 4/16	4.4258	0.000033	0.000098
Ax FA: 6/16	18.0559	0.00014	0.00038
Ax FA: 8/16	71.0475	0.00054	0.0015
Ax FA: 10/16	285.9959	0.0022	0.0062
Ax FA: 16/16	51626	0.8665	0.3939
32-bit RCA			
Ax FA: 4/32	4.3603	$<$ e-09	$<$ e-08
Ax FA: 8/32	71.8584	$<$ e-08	$<$ e-07
Ax FA: 16/32	18132	0.0000021	0.0000061
Ax FA: 24/32	4761400	0.000554	0.0015
Ax FA: 32/32	3.527e+09	0.8776	0.4105

MED, NMED, and MRED to efficiently evaluate the accuracy of the adder. A more detailed information can be found in [28]–[33]. We performed the following simulations in MATLAB with Cin='0'. We created a behavioral-level model of the RCA, which can model variable approximation degree and the number of bits.

1) Error metrics for 8-bit RCA: For the 8-bit case, we applied all input combinations to the RCAs with different approximation degrees. With this setup the MED, NMED and MRED were determined. We used the approximated full adders for the Least-Significant Bits (LSBs) of the RCA. The cases with one to five approximated full adders were recorded in Table VI. We observed that the MED and thus also the NMED roughly double per included approximated full adder.

2) Error metrics for 16-bit and 32-bit RCA: In the analysis of 16-bit and 32-bit RCA, we used one million randomly generated numbers as input variables. We did this because for a complete evaluation 2^{2n} input combinations would be required, which is computationally intensive. We again simulated a RCA with different approximation degrees and determined the error metrics which are displayed in Table VI. The results of the 16- and 32-bit simulations yield drastically reduced NMED and MRED values with the same percentage of approximated adders, if compared to the 8-bit simulation. This indicates that an approximated full adder generates a substantially higher quality output with a higher number of bits. Since only one million input combinations were validated, the results are subject to stochastic deviations. However, the current numbers should be a good representative since the one million input combinations were selected at random.

V. Circuit-Level comparison

A. Comparison with exact full adders

1) Energy consumption: We recreated the energy consumption simulation of the exact algorithm in the semi-serial topology [11] with the IMPLY specific values from Table V to allow for a fair comparison. We measured that $3.8435nJ$ is required on average per step and the extra steps together used $0.8053nJ$. If we compare the result to the exact full adder in the semi-serial structure [11], the presented algorithm is up to 34% more efficient for an 8-bit configuration (excluding any steps that are only executed once).

2) Number of steps: Another important circuit-level metric is the number of necessary steps per bit. As shown in Section III our algorithm requires 5 steps per bit and a step to reset the work memristors. The exact semi-serial algorithm from [11] needs $10n+2$ steps. If we combine both algorithms in a RCA with k approximated and $n-k$ exact full adder the number of steps for n-bit is calculated through

$$Steps(n,k) = 5k + 10(n-k) + 3. \tag{4}$$

In comparison at 8-bit, $5\% - 29\%$ fewer steps are required by our proposed algorithm, depending on the approximation degree.

3) Area usage: The third important comparison point on circuit-level is the area usage, which is determined by the number of required memristors and switches. The proposed algorithm uses $2n+3$ memristors for n-bit as well as 12 additional CMOS-switches. Since the RCA only partly consists of approximated adders, the area usage is based on the exact semi-serial algorithm from [11]. It follows that the RCA needs $2n+6$ memristors and 12 switches.

B. Comparison with approximate full adders

To give a comparison to the related SoA, we compare the presented algorithms to the approximate full adders from [34], since they also use IMPLY logic. We did not compare to other approximate adders because the disparity would be too significant to make a meaningful comparison. That is, they are based on a different type of in-/near-memory computing paradigm and logic, which would render the comparison unfair. We compared the algorithms at an approximation degree of 5/8.

1) Energy consumption: Compared to the approximated full adders from [34], the energy consumption of our algorithm is worse by at least 52%. This is due to the different topologies used. Both our approach and the SIAFA adders were able to reduce the energy consumption by half if we compare it to the respective exact algorithm for the given structure used.

2) Number of steps: The algorithm presented in this work excels in speed. The algorithm we presented requires $45-50\%$ less steps than the SIAFA algorithms. This is a significant difference, considering that both are approximated algorithms.

TABLE VII: Circuit-Level Comparison to the SoA full adder

Algorithm	Energy consumption (nJ)		No. of steps		No. of memristors		No. of switches
	n, k	n=8-bit, k=5	n, k	n=8-bit, k=5	n	n=8-bit	n
Serial Exact [13]	1.8531n	14.8248	22n	176	2n+3	19	0
Semi-Serial Exact [11]	3.8435n + 0.81	31.558	10n + 2	82	2n+6	22	12
SIAFA 1,3 [34]	0.6444k + 1.8531(n-k)	8.7813	8k + 22(n-k)	106	2n+3	19	0
SIAFA 2 [34]	0.8049k + 1.8531(n-k)	9.5838	8k + 22(n-k)	116	2n+3	19	0
SIAFA 4 [34]	0.6431k + 1.8531(n-k)	8.7748	8k + 22(n-k)	106	2n+3	19	0
Proposed	1.6678k + 3.8435(n-k) + 0.865	20.7345	5k + 10(n-k) + 3	58	2n+6	22	12

3) Area usage: Both algorithms number of memristors are in the order of $2n$ memristors for n-bit adders (ours needs 3 more memristors). The algorithm we presented requires an additional 12 CMOS switches, while the algorithms from [34] do not.

4) Error metrics: The presented algorithm has the same ER and has almost the same NMED and MRED as SIAFA1 and SIAFA3, which exhibit the best accuracy out of the algorithms presented in [34]. In comparison to SIAFA2 and SIAFA4, the proposed full adder demonstrates a superior NMED, which is lower by 34% and 16% at an approximation degree of 5/8. The same trend applies to the 16-bit and 32-bit error metrics. It should be noted that a stochastic deviation is expected in these simulations, which was explained in more detail in Section IV.

VI. Application in image processing

Image processing is an error-resistant application that is used in many different aspects of our lives such as medicine, industry, automation, robotics, and media [36]. We simulated the presented algorithm in an RCA structure using MATLAB. We assessed various levels of approximation and determined the quality metrics Peak Signal-to-Noise Ratio (PSNR) and Mean Structural Similarity Index Measure (MSSIM) [42]. The quality metrics of each application are presented in Table VIII.

A. Image addition

Image addition is one of the most basic applications of image processing and is used for masking and enhancing [31]. To simulate this, we added each pixel of the first image to the corresponding pixel of the second image and halved the result. We added two 8-bit standard images using a RCA with varying approximation degrees. With one to five approximated full adders the PSNR threshold of 30dB was surpassed, which demonstrates the acceptable image quality when using these adders. With six or more approximated adders it could not be reached. The resulting images with different approximation degrees are shown in Figure 5.

B. Image subtraction

We carried out the image subtraction, by inverting the subtracted image and representing it as 2s complement. Otherwise, the procedure is the same as with image addition. Image subtraction is often used for motion detection, robotics, medicine, and surveillance systems [36]. We took two 8-bit images from the image database of [43] as an example. The PSNR value was acceptable for up to five out of eight approximated adders, while the MSSIM deteriorates rapidly with increased approximation. The results with one, three, and five approximated full adders are presented in Figure 6.

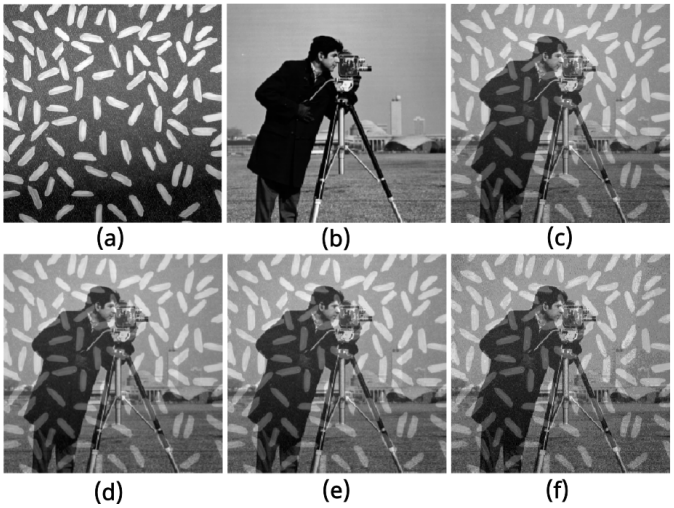

Fig. 5: Results of image addition with different approximation degrees: (a) rice, (b) cameraman, (c) exact image addition, (d) one ax adder, (e) three ax adder, (f) five ax adder

TABLE VIII: Quality metrics of image processing applications

Approximated Algorithm	Image addition		Image subtraction		Gray-scale filter	
	PSNR (dB)	MSSIM	PSNR (dB)	MSSIM	PSNR (dB)	MSSIM
Proposed (1/8 Ax FA)	51.12	0.9976	58.76	0.9974	52.90	0.9984
Proposed (2/8 Ax FA)	47.17	0.9941	51.80	0.9909	49.93	0.9970
Proposed (3/8 Ax FA)	43.31	0.9860	45.26	0.9520	45.49	0.9910
Proposed (4/8 Ax FA)	38.31	0.9603	39.45	0.8602	40.22	0.9693
Proposed (5/8 Ax FA)	32.93	0.8934	33.74	0.7183	34.05	0.9003

C. Gray-scale filter

The gray-scale filter transforms a colored RGB image into an image with only gray-scale pixels. To accomplish this the individual color values are added together in two iterations and the result is divided by three. This was simulated with different approximation degrees on a standard image. This simulation reaches the PSNR threshold of 30dB with five or fewer approximated adders in an 8-bit RCA. The resulting images with different approximation degrees are shown in Figure 7.

TABLE IX: Application level comparison to exact semi-serial adder [11]

Algorithm	Image addition (256x256 8-bit Image)				Image subtraction (512x512 8-bit Image)				Gray-scale filter (684x912 8-bit Image)			
	Energy/ pixel (nJ)	Total Energy (mJ)	Steps/ pixel	Total Steps (million)	Energy/ pixel (nJ)	Total Energy (mJ)	Steps/ pixel	Total Steps (million)	Energy/ pixel (nJ)	Total Energy (mJ)	Steps/ pixel	Total Steps (million)
Semi-Serial Exact [11]	31.558	2.068	82	5.374	31.558	8.273	82	21.496	63.116	39.372	164	102.305
Proposed (1/8 Ax FA)	29.434	1.929	78	5.112	29.434	7.716	78	20.447	58.875	36.727	156	97.314
Proposed (5/8 Ax FA)	20.735	1.359	58	3.801	20.735	5.436	58	15.204	41.469	25.869	116	72.362

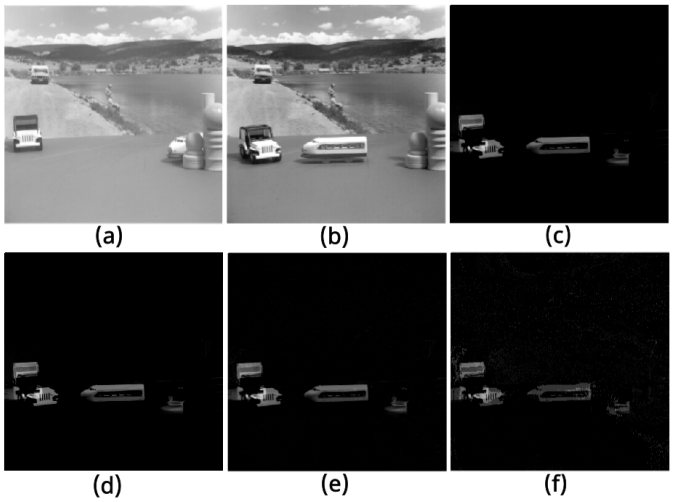

Fig. 6: Results of image subtraction with different approximation degrees: (a) first image [43], (b) second image [43], (c) exact image subtraction, (d) one ax adder, (e) three ax adder, (f) five ax adder

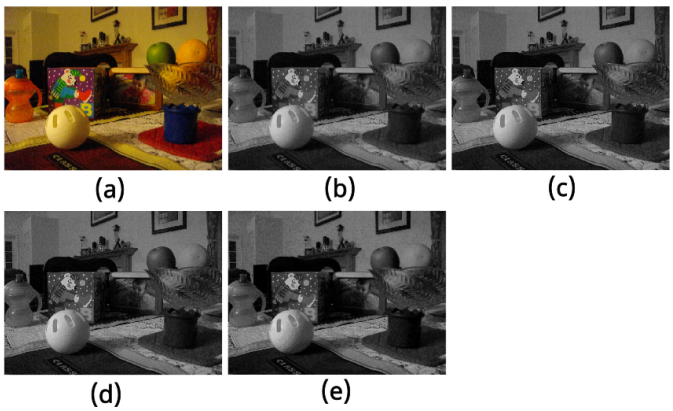

Fig. 7: Results of the gray-scale filter with different approximation degrees: (a) standard image, (b) exact gray-scale filter, (c) one ax adder, (d) three ax adder, (e) five ax adder

D. Application-level comparison

1) Using Exact Semi-Serial Adder [11]: To effectively compare our algorithm with the exact approach [11] in image processing we looked at the difference per pixel for each application. We compared the RCA structures with one and five out of eight approximated adders with the exact 8-bit RCA. The approximation degrees in between behave linearly and so were left out. If we sum up the required energy and steps per pixel, we get the total energy consumption and number of steps per image processing application. With our approach, $9\% - 37\%$ less energy and $5\% - 29\%$ fewer steps are required for the image processing applications while the accuracy is still acceptable as shown in Table VIII. We were able to save $13.5mJ$ of energy and about 30 million steps for a gray-scale filter of a 684x912 8-bit image. An overview of the improvements for image addition and subtraction is presented in Table IX.

2) Using Approximated Adder from [34]: As explained in Section V, since the topology from [34] differs from the semi-serial structure we have used, stark differences in energy consumption and number of steps are expected. This was explained more in-depth in Section V. The following comparisons were done with an approximation degree of 5/8.

In image addition, our algorithm shows the best results together with SIAFA1. In image subtraction, our algorithm shows better PSNR than SIAFA1-3 by $1dB$ but exhibits worse results than SIAFA4. At gray-scale filtering the presented algorithm demonstrates worse PSNR than SIAFA1,3 but still improves over SIAFA2,4 by at least $2.5dB$.

VII. Conclusion

In this paper, we presented an approximated algorithm that utilizes IMPLY logic in a semi-serial topology. The main focus of the approximation was to lower the required steps per bit and the energy consumption while maintaining sufficient accuracy for error-resistant applications. With our methodology, we managed to save up to 34% energy and $5\% - 29\%$ of the steps required per bit. In comparison to other IMPLY-based approximated adders at 8-bit, we were able to save $45 - 50\%$ steps with our proposed algorithm, at equal or better accuracy. We verified the functionality of our algorithm using LT-SPICE and evaluated the proposed algorithm in a RCA. We conducted a behavioral-level simulation and evaluated the error metrics using MATLAB. We did a comparison to the exact semi-serial algorithm and SoA approximated IMPLY-based adders. We applied the presented algorithm in various image processing applications and evaluated the performance of the proposed algorithm with varying approximation degrees. We showed how this method leads to large improvements in speed and energy consumption at the application level. Our results indicate that for up to five out of eight approximated adders the results of PSNR are sufficient for all applications. Further analysis of different approximations in hybrid structures and their applicability in 16-bit and 32-bit RCA and other fields are important areas for future research.

References

[1] W. Liu *et al.* A retrospective and prospective view of approximate computing. *Proceedings of the IEEE*, 108:394–399, 03 2020.

[2] V. Gupta *et al.* Impact: Imprecise adders for low-power approximate computing. In *IEEE/ACM International Symposium on Low Power Electronics and Design*, pp. 409–414, 2011.

[3] V. Gupta *et al.* Low-power digital signal processing using approximate adders. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 32(1):124–137, 2013.

[4] N. TaheriNejad and S. Shakibhamedan. Energy-aware adaptive approximate computing for deep learning applications. In *2022 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, pp. 328–328, 2022.

[5] N. Taherinejad *et al.* Memristors' potential for multi-bit storage and pattern learning. In *2015 IEEE European Modelling Symposium (EMS)*, pp. 450–455, Oct 2015.

[6] N. Taherinejad *et al.* Fully digital write-in scheme for multi-bit memristive storage. In *2016 13th International Conference on Electrical Engineering, Computing Science and Automatic Control (CCE)*, pp. 1–6, Sept 2016.

[7] D. Radakovits and N. TaheriNejad. Implementation and characterization of a memristive memory system. In *2019 IEEE 32nd Canadian Conference on Electrical and Computer Engineering (CCECE)*, pp. 1–5, May 2019.

[8] J. Borghetti *et al.* Memristive switches enable stateful logic operations via material implication. *Nature*, 464:873–6, 04 2010.

[9] E. Lehtonen and M. Laiho. Stateful implication logic with memristors. *2009 IEEE/ACM International Symposium on Nanoscale Architectures*, pp. 33–36, 2009.

[10] C. Li *et al.* In-memory computing with memristor arrays. In *2018 IEEE International Memory Workshop (IMW)*, pp. 1–4, 2018.

[11] N. TaheriNejad *et al.* A semi-serial topology for compact and fast imply-based memristive full adders. In *2019 17th IEEE International New Circuits and Systems Conference (NEWCAS)*, pp. 1–4, 2019.

[12] L. Chua. Memristor-the missing circuit element. *IEEE Transactions on Circuit Theory*, 18(5):507–519, 1971.

[13] S. G. Rohani and N. TaheriNejad. An improved algorithm for imply logic based memristive full-adder. In *2017 IEEE 30th Canadian Conference on Electrical and Computer Engineering (CCECE)*, pp. 1–4, 2017.

[14] D. Radakovits *et al.* A memristive multiplier using semi-serial imply-based adder. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 67(5):1495–1506, 2020.

[15] S. E. Fatemieh *et al.* Approximate in-memory computing using memristive imply logic and its application to image processing. In *2022 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 3115–3119, 2022.

[16] A. Karimi and A. Rezai. Novel design for a memristor-based full adder using a new imply logic approach. *Journal of Computational Electronics*, 17, 09 2018.

[17] N. TaheriNejad. Sixor: Single-cycle in-memristor xor. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 29(5):925–935, 2021.

[18] K. A. Ali. *New design approaches for flexible architectures and in-memory computing based on memristor technologies*. PhD thesis, IMT Atlantique, 2020.

[19] M. R. Alam *et al.* Exact stochastic computing multiplication in memristive memory. *IEEE Design Test*, pp. 1–8, 2021.

[20] M. R. Alam *et al.* Sorting in memristive memory. *ACM Journal on Emerging Technologies in Computing Systems*, pp. 1–22, 2022.

[21] M. R. Alam *et al.* Stochastic computing in beyond von-neumann era: Processing bit-streams in memristive memory. In *IEEE Transactions on Circuits and Systems II: Express Briefs (TCAS-II)*, volume 65, pp. 2423–2427, 2022.

[22] S. Gupta *et al.* Felix: Fast and energy-efficient logic in memory. In *2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 1–7, 2018.

[23] S. Kvatinsky *et al.* Magic—memristor-aided logic. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 61(11):895–899, 2014.

[24] S. Kvatinsky *et al.* Mrl — memristor ratioed logic. In *2012 13th International Workshop on Cellular Nanoscale Networks and their Applications*, pp. 1–6, 2012.

[25] S. Kvatinsky *et al.* Memristor-based material implication (imply) logic: Design principles and methodologies. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 22(10):2054–2066, 2014.

[26] S. Kvatinsky *et al.* Memristor-based imply logic design procedure. In *2011 IEEE 29th International Conference on Computer Design (ICCD)*, pp. 142–147, 2011.

[27] S. Ganjeheizadeh Rohani *et al.* A semiparallel full-adder in imply logic. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 28(1):297–301, 2020.

[28] S. E. Fatemieh *et al.* Lahaf: Low-power, area-efficient, and high-performance approximate full adder based on static cmos. *Sustain. Comput. Informatics Syst.*, 30:100529, 2021.

[29] H. Jiang *et al.* A review, classification, and comparative evaluation of approximate arithmetic circuits. *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, 13:1 – 34, 2017.

[30] H. Jiang *et al.* Approximate arithmetic circuits: A survey, characterization, and recent applications. *Proceedings of the IEEE*, 108(12):2108–2135, 2020.

[31] H. A. Almurib *et al.* Inexact designs for approximate low power addition by cell replacement. In *2016 Design, Automation Test in Europe Conference Exhibition (DATE)*, pp. 660–665, 01 2016.

[32] S. E. Fatemieh and M. R. Reshadinezhad. Power-efficient, high-psnr approximate full adder applied in error-resilient computations based on cntfets. In *2020 20th International Symposium on Computer Architecture and Digital Systems (CADS)*, pp. 1–5, 2020.

[33] Z. Yang *et al.* Transmission gate-based approximate adders for inexact computing. pp. 145–150, 07 2015.

[34] S. E. Fatemieh *et al.* Fast and compact serial imply-based approximate full adders applied in image processing. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 13(1):175–188, 2023.

[35] Z. Yang *et al.* Approximate xor/xnor-based adders for inexact computing. *2013 13th IEEE International Conference on Nanotechnology (IEEE-NANO 2013)*, pp. 690–693, 2013.

[36] Y. S. Mehrabani *et al.* A novel high-speed, low-power cntfet-based inexact full adder cell for image processing application of motion detector. *J. Circuits Syst. Comput.*, 26:1750082:1–1750082:15, 2017.

[37] S. Muthulakshmi *et al.* Memristor augmented approximate adders and subtractors for image processing applications: An approach. *AEU - International Journal of Electronics and Communications*, 91, 05 2018.

[38] S. Muthulakshmi *et al. Memristor-Based Approximate Adders for Error Resilient Applications*, pp. 51–59. 01 2018.

[39] K. Bickerstaff and E. E. Swartzlander. Memristor-based arithmetic. In *2010 Conference Record of the Forty Fourth Asilomar Conference on Signals, Systems and Computers*, pp. 1173–1177, 2010.

[40] S. Kvatinsky *et al.* Vteam: A general model for voltage-controlled memristors. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 62(8):786–790, 2015.

[41] D. R. M. Jungwirth and N. TaheriNejad. Spice implementation of vteam model, 2018.

[42] Z. Wang *et al.* Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.

[43] Usc university of southern california, signal and image processing institut (sipi).