International Journal of
DATA SCIENCE AND IOT MANAGEMENT SYSTEM
ISSN: 3068-272X www.ijdim.com Original Research Paper

ACCELERATED IMAGE PROCESSING THROUGH IMPLY-BASED NOCARRYAPPROXIMATED ADDERS

Afraa Anwar[1], Nilofer[2]

[1]PG Scholar, Department of VLSI, Shadan Women's College of Engineering and Technology, Hyderabad, afraa.anwar03@gmail.com

[2]Asst. Professor, Department of ECE, Shadan Women's College of Engineering and Technology nilofer.eng@gmail.com

Received: 02-08-2025 Accepted: 04-09-2025 Published: 11-09-2025

ABSTRACT

Conventional approaches to meeting the demands of computing power are finding it difficult to keep up with the sharp rise in demand. Alternative computer paradigms have therefore proliferated in an effort to address this discrepancy. An emerging technique for increasing speed, space efficiency, and energy consumption in error-resilient applications like computer vision and machine learning is approximate computing, or AxC.

Accuracy is sacrificed in exchange for these improvements. Because of their low power consumption and intrinsic non-volatility, which make them appropriate for In-Memory Computation (IMC), memristors have attracted a lot of attention from a technological standpoint. In order to address the discrepancy between performance progress and demand increase, another computer paradigm has emerged.

We use Material Implication (IMPLY), a memristive stateful in-memory logic, in this study. In the framework of AxC, we study sophisticated adder topologies with the goal of fusing the advantages of both cutting-edge computing paradigms. For every adder topology based on IMPLY, we provide two estimated methods. Compared to the comparable exact full adders, they lower the number of steps by 6% to 54% and the energy consumption by 7% to 54% when integrated into a Ripple Carry Adder (RCA).We compare our work with State-of-the-Art (SoA) circuit-level approximations that improve speed and energy efficiency by up to 72% and 34%, respectively, and lower the Normalized Median Error Distance (NMED) by up to 81%. We assess our adders in four widely used image processing applications and give two more test datasets. In most cases, our proposed adders may reduce the number of image processing steps and energy usage by up to 60% and 57%, respectively, while improving quality metrics over the SoA.

I. INTRODUCTION

Since a large percentage of fundamental instructions rely on addition and multiplication, adding operations are fundamental to digital arithmetic [1]. Improving adders is essential to raising total computing performance in order to satisfy the quickly increasing demand for processing power. As Moore's Law slows down [2], transistors hit their physical limits [3], and computing's footprint grows exponentially [4], more emphasis is being paid to investigating novel computing paradigms and cutting-edge technology. AxC is quickly becoming a viable way to improve compute efficiency and solve the power-wall issue [2], [5]. It is possible to obtain notable improvements in speed, area, and energy consumption by approximating computer operations. These enhancements come at the expense of accuracy [1], [2], and [5].

Approximating some features can greatly reduce computer time and power consumption since image and video processing applications are error robust [2], [6], and [7]. Furthermore, sectors that are closely related to imaging applications, such robotics, data mining, communication, pattern recognition, and machine learning, might also benefit [2], [8], [9], [10], and [11].Complementary Metal-Oxide Semiconductor (CMOS) technology has been used to create a number of estimated adders [1], [12], [13], and [14].The Von-Neumann bottleneck, which normally arises between logic and memory, is the fundamental issue that unites them all. One possible remedy for this problem is IMC, which is a method for carrying out calculations directly in memory. One noteworthy new component that shows promise is the memristor [15]. The memristor is the best option for IMC memory cells because of its intrinsic capacity to carry out logical processes and store non-volatile data in its resistive state [16], [17]. Memristors are further positioned as prospective candidates for

International Journal of
DATA SCIENCE AND IOT MANAGEMENT SYSTEM
ISSN: 3068-272X www.ijdim.com Original Research Paper

future computing breakthroughs by additional characteristics including their compact form factor and low power consumption [18], [19], [20], and [21]. The stateful logic IMPLY is one of the most widely used options in the context of IMC. It proved to be the most dependable stateful logic in [22] and is compatible with the crossbar array.

believing it to be the best option for these kinds of applications [16], [23]. There are three types of structures that are now available for carrying out IMPLY operations: serial, parallel, and hybrid.[19], [24], [25], and [26] topologies. Every topological implementation is competitive because it provides unique benefits in one or more criteria, such speed or space utilization. Two estimated IMPLY-based adders for the serial, parallel, semi-serial, and semi-parallel topologies are shown in this study.

PROPOSED METHOD

The following is a summary of this paper's contributions:

• Eight new approximated adders were created using two novel approximation techniques for imply-based adder design for each of the four imply-based topologies, enhancing speed, energy consumption, and error metrics in comparison to soa exact and approximated adders;

• Offering a new evaluation dataset for grayscale filtering and image addition;

• For the first time, introducing estimated adders to the parallel and semi-parallel architecture.

II. LITERATURE REVIEW

A. Raghunathan, S. P. Park, D. Mohapatra, K. Roy, and V. Gupta The impact Accurate adders for approximation computation with reduced power consumption Portable multimedia devices that use different signal processing techniques and architectures must have low power consumption. Human senses, which are not flawless, interpret the end product in the majority of multimedia applications. The requirement to provide accurate numerical results is eliminated by this fact. Prior studies in this area take use of error-resiliency mostly by voltage over-scaling, with the mistakes that follow being reduced by computational and architectural methods. As an alternate strategy to capitalize on the relaxation of numerical precision, we suggest logic complexity reduction in this study. We illustrate this idea by putting forward a number of approximate or imprecise Full Adder (FA) cells that are simpler at the transistor level. We then use these cells to create approximate multi-bit adders. Apart from the intrinsic decrease in switching capacitance, our methods lead to noticeably shorter critical pathways, which facilitate voltage scaling. Using the suggested approximate arithmetic units, we create designs for image and video compression algorithms and test them to show how effective our method is. When compared to current implementations, post-layout simulations show power and area reductions of up to 60% and 37%, respectively, with no output quality loss.

M. Schulte, F. Lombardi, and W. Liu, A look into approximation computing from both a past and a future perspective Traditionally, computing systems are built to function as precisely as feasible. However, there are significant technological obstacles to this trend, including high performance, circuit dependability, and power consumption. Computing system performance and power consumption have been continuously increased for about 50 years, mostly through technological scaling. According to Dennard's scaling, transistors have become more smaller and their supply voltage has decreased over time, allowing circuits to function at greater frequencies while dissipating almost the same amount of power. However, it is challenging to enhance performance further under the same power limitations since Dennard's scaling trends toward an end. Power consumption has always been a significant issue and is currently a crucial issue for the whole industry.When the feature size of complementary metal-oxide-semiconductor (CMOS) technology is decreased below 7 nm, reliability declines in addition to power because it becomes more difficult to manage and avoid defects and parameter fluctuations at advanced nanoscales. As a result, production and verification costs will rise dramatically to guarantee the total correctness of signals, logic values, devices, and interconnects.

V. Gupta, D. Mohapatra, A. Raghunathan, and K. Roy, Approximate adders for low-power digital signal processing For portable multimedia devices using different signal processing methods and architectures, low power consumption is a crucial need. Humans can infer valuable information from somewhat inaccurate outputs in the majority of multimedia applications. As a result, we are not required to generate accurate numerical results.

International Journal of
DATA SCIENCE AND IOT MANAGEMENT SYSTEM
ISSN: 3068-272X www.ijdim.com Original Research Paper

In this setting, earlier research takes use of error resilience.mostly by voltage over scaling, with the ensuing mistakes being mitigated through computational and architectural approaches. In order to exploit the relaxation of numerical precision, we suggest in this study a different strategy: logic complexity reduction at the transistor level. In order to illustrate this idea, we suggest a number of approximate or imprecise complete adder cells that are simpler at the transistor level. We then use these cells to create approximation multi-bit adders. Apart from the intrinsic decrease in switching capacitance, our methods lead to noticeably shorter critical pathways, which facilitate voltage scaling. Using the suggested approximate arithmetic units, we create designs for image and video compression algorithms and test them to show how effective our method is. Additionally, we establish basic mathematical models for these approximation adders' inaccuracy and power consumption. Additionally, we show how useful these approximation adders are in two digital signal processing architectures with particular quality constraints: the finite impulse response filter and the discrete cosine transform. Comparing the suggested approximation adders to current implementations that use correct adders, simulation findings show power savings of up to 69%.

N. Taheri Nejad and F. Seiler, A semi-serial approximation in-memristor adder based on IMPLY In recent years, research and development has focused heavily on new technologies and computing paradigms to help ease the Von Neumann bottleneck. From a computational and technical standpoint, memristors provide novel opportunities. Because they can carry out logical operations in memory, they are appropriate for In-Memory Computation (IMC) and have good data storage capabilities. Approximate computing, which is employed in error-resistant applications, is another new computing paradigm that lowers computation time and space usage.

Here, we suggest a brand-new approximated complete adder that employs a semi-serial structure and the stateful logic Material Implication (IMPLY). We include this complete adder into a Ripple Carry Adder (RCA), which we then assess at the circuit level. The error metrics were assessed and contrasted with adders based on State-of-the-Art (SoA) IMPLY. In comparison to the precise technique, our solution uses up to 29% fewer steps and up to 34% less energy at 8-bit, and the Normalized Median Error Distance (NMED) is less than 0.01 in the majority of cases. The corresponding quality metrics are computed after applying the suggested adder to image processing. Given that the Peak Signal-to-Noise Ratio (PSNR) is more than 30 dB, all tested approximation degrees produce a suitable outcome. In comparison to the precise calculations, the suggested method allows us to save almost 13.5mJ of energy while gray-scale filtering a 684×912 8-bit picture.

A. S. Baroughi, H. S. Shahhoseini, N. Taheri Nejad, N. Amirafshar, and S. Shakibhamedan, ACE-CNN: Energy-efficient CNN-based image categorization with approximate carry disregard multipliers The Signed Carry Disregard Multiplier (SCDM8) is a series of signed approximation multipliers designed specifically for Convolutional Neural Network (CNN) integration. To assess the trade-off between accuracy and approximation, extensive tests were carried out on well-known pre-trained CNN models, such as VGG16, VGG19, ResNet101, ResNet152, MobileNetV2, InceptionV3, and ConvNeXt-T. The outcomes show that ACE-CNN works better than other setups, providing a favorable trade-off between accuracy and computational economy. According to our tests, SCDM8 reduces power usage by 35% on average while only slightly lowering accuracy by 1.5% when used with VGG16. Similarly, SCDM8 saves 42% of the energy while only compromising 1.8% of the accuracy when integrated into ResNet152. The first approximation version of ConvNeXt, ACE-CNN, offers an energy improvement of up to 72% at a cost of less than 1.3% Top-1 accuracy. These findings demonstrate how well SCDM8 works as an approximation technique for a range of CNN models. For image classification tasks in CNNS, our investigation demonstrates that the ACE-CNN performs better than state-of-the-art methods in terms of accuracy, energy efficiency, and computing precision. When we looked at how resilient CNN models were to approximate multipliers, we found that resnet101 was the most resilient, with an average accuracy difference of 0.97%, while lenet5 Inspired-CNN was the least resilient, with an average accuracy difference of 2.92%. By providing an efficient approximation method for CNN multipliers, our findings help choose energy-efficient approximate multipliers for CNN-based systems and advance the creation of energy-efficient deep learning systems.

International Journal of
DATA SCIENCE AND IOT MANAGEMENT SYSTEM
ISSN: 3068-272X www.ijdim.com Original Research Paper

Significant energy reductions with almost no loss of accuracy are made possible by the proposed SCDM8 family of approximation multipliers, which opens up new possibilities for effective deep learning applications.

H. Chible, M. Saleh, M. Alameh, M. Osta, M. Ibrahim, and M. Valle, Techniques for approximate computation in embedded machine learning Incorporating intelligence into modern application areas like wearable technology, portable healthcare systems, and the Internet of Things is made possible by embedding machine learning. An evaluation of approximation computing techniques at the algorithmic, architectural, and circuit levels is presented in this work along with suggestions for future advancements and uses. The primary objective Aims to look at how approximate computing could make embedded Machine Learning (ML) systems more feasible and less complex. Even though machine learning (ML) is a strong paradigm for applications in the perceptual domain (vision, touch, hearing, etc.), real-time operation and ultra-low power are still highly difficult goals because of their enormous computational complexity. However, approximation computing has become a viable way to lower time delay, simplify hardware, and boost energy efficiency.

Bio-inspired imprecise computational blocks for effective VLSI implementation of soft-computing applications, H. R. Mahdiani, A. Ahmadi, S. M. Fakhraie, and C. Lucas To calculate the exact outcomes of the given computations, traditional digital hardware computational blocks with various architectures are used. Our suggested Bio-inspired Imprecise Computational blocks (bics) are unique in that they are made to deliver a useful approximation of the outcome rather than its exact value at a reduced cost. Compared to its exact competitors, these new structures are more efficient in terms of area, speed, and power usage. This work introduces the synthesis findings, error behaviors, and detailed descriptions of example BIC adder and multiplier architectures. The hardware defuzzification block of a fuzzy processor and a three-layer face recognition neural network are then demonstrated to be effectively implemented using these BIC structures.

Inexact designs for estimated low power addition by cell replacement by H. A. Almurib, T. N. Kumar, and F. Lombardi Three designs of an approximation computing inexact adder cell are proposed in this research. Comparing these cells to both known inexact designs and an exact complete adder cell, a significantly less number of transistors are needed. At 45 nm, these imprecise cells are simulated and compared in terms of error metrics (like error rate) and circuit-based metrics (like energy consumption, latency, complexity, and energy delay product). Image addition is then explored as an application after evaluating several metrics for approximation computing through extensive simulation by substituting inexact cells, as those suggested in this work, for exact cells in a ripple carry adder. These findings demonstrate that the suggested designs outperform the current inexact cells reported in the technical literature in terms of latency, switching capacitance, and error metrics for picture quality and processing, while also using the least amount of power.

III. PROPOSED METHODOLOGY

3.1 METHODOLOGY

Redefining approximation logic functions based on precise logic is a key strategy in the construction of approximated circuits [2]. This can be accomplished by employing a modified truth table or by removing or altering parts of the precise circuit [2], [38]. We can only utilize IMPLY and FALSE operations since we are working on topologies that are based on IMPLY. It is possible to simulate Boolean logic with just these two functions as they together make up the entire logic set {→, ⊥} [27], [47]. One IMPLY operation, denoted as a = a → 0, can be used to simulate an inversion. Another memristor that has been previously reset is necessary for this. The only functions that can be replicated using two IMPLY operations are OR and NAND, which makes them excellent candidates for approximations. To minimize the ER for Sum and Cout while reducing the number of steps, the state-of-the-art (SOA) IMPLY-based approximations were created [6, 34, 35].

$$C_{out,i}^{NC} = 0$$
$$Sum_i^{NC} = a_i + b_i = \overline{a_i} \rightarrow b_i.$$

Our goal in this study is to design the quickest adder while maintaining acceptable quality, not to decrease the ER. We developed and put into practice two IMPLY-based algorithms for various adder topologies, drawing inspiration from the methodology from [12]. As a result, we are using

International Journal of
DATA SCIENCE AND IOT MANAGEMENT SYSTEM
ISSN: 3068-272X www.ijdim.com Original Research Paper

the effective OR emulation, which just needs two IMPLY operations. For the parallel, semi-parallel, serial, and semi-serial structures, we developed two algorithmic strategies. For the remainder of this study, we will refer to the algorithms by their topology in conjunction with either NoCarry+ for the advanced implementation or IMPLY-based NoCarry (NC) for the base. Since the carry-out is not propagated (for example, SINC and SINC+ for the Serial IMPLY-based NoCarry(+) algorithm), this name was selected. To establish the Sum for each bit, the simple version just uses OR combinations between the a and b inputs. No carry-out is created or spread, and the carry-in is totally ignored. The Cout is therefore assigned to the logical value "0." The NoCarry version's logical equations are

By altering the final approximated bit, we improved the basic version in our second implementation. Up to the final estimated complete adder, we use the same formulae to compute the Sum and Cout. We still ignore the carry-in at this point, but we create a carry-out instead.to spread. This time, we set the Cout to be an AND b in order to reduce the likelihood of spreading the fake Cout to the precise bits. For this enhanced implementation, the logical equations are using an RCA integrated with k approximation and n − k precise adders. To ensure compatibility with precise adders, we made sure that the sum for each suggested method is kept in the b-memristor. Table II displays the truth table for both the standard and sophisticated implementations, with the incorrect locations highlighted in red. While the ER of Cout is 4/8 for the NoCarry and only 2/8 for the sophisticated NoCarry+ adder, the ER of Sum is 4/8 for both implementations.

$$C_{out,i}^{NC+} = \begin{cases} 0 & i < k \\ a_i b_i = \overline{a_i \rightarrow \overline{b_i}} & i = k \end{cases}$$

$$Sum_i^{NC+} = a_i + b_i = \overline{a_i} \rightarrow b_i,$$

The carrying is the Cout of the preceding bit as we wish to incorporate our whole adder as the bottom bits in an RCA. In the advanced form, the carry-in is likewise "0" since the Cout is set to "0" for all adders except the final one. This missing carry propagation attribute allows us to simplify the truth table to a form in which c = 0 and only a and b can change. Table III, where the ER of Sum is lowered to 1/4, displays the reduced truth table. The NoCarry+ adder has no erroneous Cout position in the reduced truth table, although the NoCarry adder's ER of Cout is likewise lowered to 1/4. With this simplified version, an error only happens when both a and b are logical "1." Here, it is crucial to remember that the truth table is not actually decreased. The inputs that can occur at each estimated bit are highlighted using this reduced truth table as a visualization tool.

TABLE 3.1: NOCARRY AND NOCARRY+ TRUTH TABLE

Inputs			Exact		No Carry		No Carry +	
a	b	c	Sum	Cout	Sum	Cout	Sum	Cout
0	0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	0	0
0	1	0	1	0	1	0	1	0
0	1	1	0	1	1	0	1	0
1	0	0	1	0	1	0	1	0
1	0	1	0	1	1	0	1	0
1	1	0	0	1	1	0	1	1
1	1	1	1	1	1	0	1	1

TABLE 3.2: REDUCED NOCARRY AND NOCARRY+ TRUTH TABLE

Inputs			Exact		No Carry		No Carry +	
a	b	c=0	Sum	Cout	Sum	Cout	Sum	Cout
0	0	0	0	0	0	0	0	0
0	1	0	1	0	1	0	1	0
1	0	0	1	0	1	0	1	0
1	1	0	0	1	1	0	1	1

A Serial Topology Consisting of SINC/SINC+

Memristors arranged in the same row or column of a crossbar array and linked to the same resistor make up the serial IMPLY topology [16], [27]. Because there is just one processing region, activities are carried out sequentially and parallelization is not feasible. Table IV displays the SINC algorithm in its entirety. To ensure proper operation, we reset the work memristor in the first stage. We compute and store the inversion of an in the work memristor in the second phase. The Sum result, which is a + b, is then saved in the b-memristor. Table V, which follows the same methodology as the previous version, shows the precise steps involved in the SINC+ algorithm. The operations indicated in blue, which are only calculated once in the final estimated adder, are where they diverge. As a result, we require a second work memory. Prior to the Sum being stored in the b-memristor in the third phase, it is utilized to save the inversion of b. The inversion is then stored in the c-memristor once ab has been calculated. As a result, the Cout is equal to ab and is usable for higher-bit computation. For an n-bit calculation, the SINC method requires 2n + 1

International Journal of
DATA SCIENCE AND IOT MANAGEMENT SYSTEM
ISSN: 3068-272X www.ijdim.com Original Research Paper

memristors and 3n steps. As we shall see in subsequent sections, the SINC+ algorithm has improved error-reduction behavior but requires 3n + 3 steps and 2n + 2 memristors.

TABLE 3.3: EXACT SINC ALGORITHM PROCEDURE IN SERIAL TOPOLOGY

Steps	Operation	Equivalent Logic
1	$w_1 = 0$	$False(w_1)$
2	$w_j^{'} = a \rightarrow w_1$	$w_1 = \overline{a}$
3	$b^{'} = w_1^{'} \rightarrow b$	$b = \overline{a} \rightarrow b = a + b = Sum$

B. PINC/PINC+ parallel topology

There are n distinct serial topologies that make up the parallel topology. CMOS switches can be used to link each of them to a common c-memristor [19]. Every bit in this structure may be calculated concurrently. Because they depend on the result of the previous bit, the only steps that cannot be completed at the same time are those that depend on the carry-in. The serial implementations from Table IV and Table V follow the same precise process as PINC and PINC+. Once more, just the final bit of the PINC+ version uses the procedures in blue.

TABLE 3.4The precise steps used by the SINC+ algorithm in serial topology
(BLUE-COLORED PERFORMANCES ARE ONLY DONE AT THE LAST APPROXIMATED BIT OF SINC+.)

Steps	Operation	Equivalent Logic
1	$w_1 = 0, w_2 = 0$	$False(w_1, w_2)$
2	$w_1^{'} = a \rightarrow w_1$	$w_1 = \overline{a}$
-	$w_2 = b \rightarrow w_2$	$w_2 = \overline{b}$
3	$b^{'} = w_1^{'} \rightarrow b$	$b = \overline{a} \rightarrow b = a + b = Sum$
-	$w_2^{''} = a \rightarrow w_2^{'}$	$w_2 = \overline{ab}$
-	$c^{'} = w_2^{''} \rightarrow c$	$c = a \rightarrow \overline{b} = ab = C_{out}$

However, we compute all of the bits at once rather than one after the other. We can fully parallelize each estimated bit since the PINC and PINC+ algorithms are independent of the carry-in. Twelve carry-independent steps are initially calculated by the precise parallel method from [19]. Therefore, before the carry-dependent portion of the first exact algorithm starts, both PINC and PINC+ are run. By using the PINC+ technique for the lower bits in an RCA setup, we were able to display this in Figure 3. As we can see, the number of precise adders n − k is the sole factor that affects the overall number of steps. The PINC algorithm is no different. For an n-bit addition, the PINC method only needs three steps and three memristors. The PINC+ adder requires 3n + 1 memristors and 6 stages.

C. S-SINC/S-SINC+ is the semi-serial topology.

The a and b inputs are arranged in two parallel rows in the semi-serial topology [25]. Switches can be used to link both rows to the work and carry memristors. In accordance with the original paper [25], we will refer to the rows containing the a-memristors and the b-memristors as Section I and Section II. Table VI displays the S-SINC method, where we decreased the number of steps needed perreduced to two.

This was accomplished using a work memristor switching system, in which the calculations are made using w1 and w2 in turn. Parallel to this, the unused work memristor is reset. Both work memristors must be reset by taking an extra step before the first bit.

Table VII displays the S-SINC+ algorithm. By adding two more stages to the S-SINC method, we compute the Cout = ab. The two work memristors are utilized in parallel for the last estimated bit. For a nbit computation, S-SINC needs 2n + 1 steps, whereas the S-SINC+ method requires 2n + 3 steps. Four switches and 2n+2 memristors are used in the S-SINC method. Two more switches and an extra memristor are required for the Cout in the advanced method (S-SINC+).

TABLE 3.5: S-SINC ALGORITHM EXACT PROCEDURE IN THE SEMI-SERIAL TOPOLOGY
(BLUE-COLORED OPERATIONS ARE ONLY PERFORMED ONCE.)

Steps	Section 1	Section 2	Equivalent Logic
-		$w_1 = w_2 = 0$	$False(w_1, w_2)$
switch w_j with w_k $(w_1 \Leftrightarrow w_2)$, $w_j := w_1$, $w_k := w_2$			
1	$w_j^{'} = a \rightarrow w_j^{'}$	$w_k = 0$	$w_j = \overline{a}$ & $False(w_k)$
2		$b^{'} = w_j^{'} \rightarrow b$	$b = \overline{a} \rightarrow b = Sum$

TABLE 3.6: S-SINC+ ALGORITHM EXACT PROCEDURE IN THE SEMI-SERIAL TOPOLOGY
Only one calculation is made for red operations.
BLUE-COLORED OPERATIONS ARE ONLY DONE AT THE LAST APPROXIMATED BIT

Steps	Section 1	Section 2	Equivalent Logic
-		$w_1 = w_2 = 0$	$False(w_1, w_2)$
switch w_j with w_k $(w_1 \Leftrightarrow w_2)$, $w_j := w_1$, $w_k := w_2$			
1	$w_j^{'} = a \rightarrow w_j^{'}$	$w_k = 0$	$w_j = \overline{a}$ & $False(w_k)$
-		$w_k^{'} = b \rightarrow w_k$	$w_k = \overline{b}$
2	$w_k^{''} = a \rightarrow w_k^{'}$	$b^{'} = w_j^{'} \rightarrow b$	$b = \overline{a} \rightarrow b = Sum$ & $w_k = a \rightarrow \overline{b}$
-	$c^{'} = w_k^{''} \rightarrow c$		$c = a \rightarrow \overline{b} = ab = C_{out}$

International Journal of
DATA SCIENCE AND IOT MANAGEMENT SYSTEM
ISSN: 3068-272X www.ijdim.com Original Research Paper

D. S-PINC/S-PINC+ Semi-Parallel Topology

Two parallel rows (parts) with either the input memristors a and b make up the semi-parallel topology [24]. A work memristor is also included in each area. Furthermore, the The circuit's Section II contains the memristor. Switches (S1 for segment I and S3 for Section II) can be used to link each segment of the circuit to a resistor. The sections can be linked together to share data using a third switch (S2). This will be referred to as a "between sections" computation. Please refer to [24] for more specific details on the semi-parallel topology. Table VIII displays the S-PINC algorithm, which is somewhat comparable to SINC. The final step is calculated differently for each segment. For an n-bit calculation, this method needs 2n+1 Cmemristors and 3n steps. To save the inversion of b in parallel with the S-PINC steps, we employ an extra work memristor in the S-PINC+ algorithm. Next, we must compute ab and store it in the c-memristor in two phases. This means that for an n-bit addition, the method requires 2n + 3 memristors and 3n + 2 steps. Table IX shows the precise process with the switch states.

TABLE 3.7: PRECISE S-PINC PROCEDURE IN THE SEMI-PARALLEL TOPOLOGY

Steps	Operation	Section	(S1,S2,S3)	Equivalent Logic
1	$w_1 = 0$	a-Section	(1,0,0)	$False(w_1)$
2	$w_1^{'} = a \rightarrow w_1$	a-Section	(1,0,0)	$w_1 = \overline{a}$
3	$b = w_1^{'} \rightarrow b$	between	(0,1,1)	$b = \overline{a} \rightarrow b = Sum$

TABLE 3.8: EXACT PROCEDURE OF S-PINC+ IN THE SEMI-PARALLEL TOPOLOGY

(Only the final approximate bit is used for operations that are colored blue.)

Steps	Section 1	Section 2	Between sections	(S1,S2,S3)	Equivalent Logic
1	$w_1 = 0$	$w_2 = 0$	-	(1,0,0) (1,0,1)	$False(w_1)$ & $False(w_2)$
2	$w_1^{'} = a \rightarrow w_1$	$w_2^{'} = b \rightarrow w_2$	-	(1,0,0) (1,0,1)	$w_1 = \overline{a}$ & $w_2 = \overline{b}$
3	-	-	$b^{'} = w_1^{'} \rightarrow b$	(0,1,1)	$b = \overline{a} \rightarrow b = Sum$
-	-	-	$w_1^{'} = a \rightarrow w_1^{'}$	(1,1,0)	$w_1 = ab$
-	-	$c^{'} = w_1^{'} \rightarrow c$	-	(0,0,1)	$c = a \rightarrow b = ab = C_{out}$

3.2 SIMULATING AT THE CIRCUIT LEVEL:

A. Configuring a Circuit Simulation

We used LT-SPICE to simulate the algorithms at the circuit level in order to confirm their operation. We employed a model for this that was based on the SPICE implementation of the Voltage-controlled Threshold Adaptive Memristor (VTEAM) model [36], [25], [48]. By fitting the model to a genuine discrete Knowm memristor, the parameters in this model are established similarly to those in Table X [49]. This makes it easier for us to compare our work with that of others who have used the same model and boosts our confidence in the usefulness of our circuit simulations. Similar to the distinctions between integrated and discrete CMOS devices, discrete memristors operate more slowly and use more power. It becomes sense to anticipate that the integrated memristor devices would operate with much greater speed and power efficiency. We employ measurement-fitted models to guarantee a realistic and useful implementation of our suggested circuits because integrated memristors are not readily available. To enable an honest and straightforward comparison to SoA works like [6], [19], [27], and [35], we selected the IMPLY specific parameters.

4. RESULTS & DISCUSSION

The simulation and implementation of accelerated image processing using IMPLY-based no-carry approximated adders is a structured process carried out within the **Xilinx Vivado Design Suite**, a leading platform for FPGA-based design. The methodology begins by creating a new RTL (Register Transfer Level) project in Vivado, where the source files defining the arithmetic architecture are introduced. In this study, the **Kogge-Stone Adder (KSA)** is employed due to its parallel prefix structure, which enables faster computation and reduced delay compared to conventional ripple-carry or carry-lookahead adders. Its efficiency makes it particularly suitable for high-performance applications such as real-time image processing.

Following the inclusion of Verilog modules, the FPGA board specifications are defined through Xilinx's device library, ensuring correct mapping of I/O pins. At this stage, **the tb_KSA64** module is designated as the top-level source for the initial simulation. Running the simulation provides insights into device utilization, particularly the estimated versus available pins, and also generates a schematic representation of the design. These outputs assist in validating the logical structure before hardware implementation.

The subsequent step involves **implementation**, where the toolchain evaluates hardware-level performance metrics. This includes estimation of **power consumption, total on-chip power dissipation, and junction temperature**, which are

International Journal of
DATA SCIENCE AND IOT MANAGEMENT SYSTEM
ISSN: 3068-272X www.ijdim.com Original Research Paper

critical parameters in optimizing FPGA-based accelerators for energy-efficient image processing. The implementation phase also produces a summary of resource utilization, which is essential for determining scalability and feasibility on specific FPGA boards.

Next, **RTL analysis** elaborates the device architecture, offering a hierarchical view of the design. This step is particularly significant from a research perspective, as it validates whether the structural optimization achieved through approximated adders translates into reduced complexity without compromising functional accuracy. Finally, **the tb_KSA64 (Test.v)** testbench is selected as the top module, and a comprehensive simulation is executed. The final simulation output verifies functional correctness, thereby demonstrating how IMPLY-based no-carry approximated adders can accelerate image processing by achieving a balance between performance, power efficiency, and hardware resource utilization.

This methodology not only validates the feasibility of approximate arithmetic in hardware accelerators but also highlights its potential for applications requiring high-speed and energy-efficient image processing.

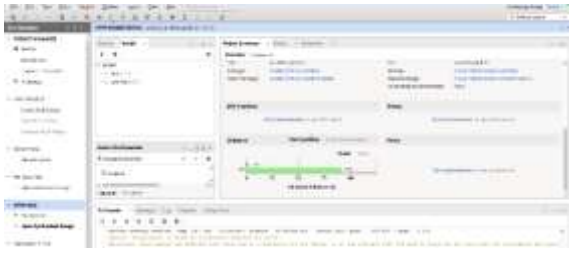

FIGURE: SIMULATION OF THE PROJECT
(the utilization of the I/O pins)

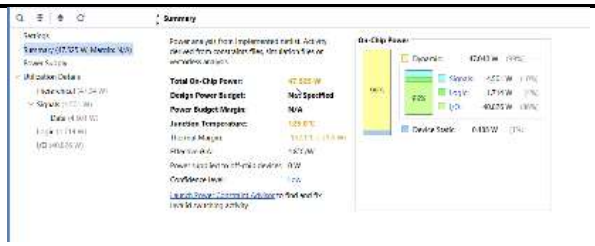

FIGURE: IMPLEMENTATION OF THE PROJECT
(on-chip power consumption details)

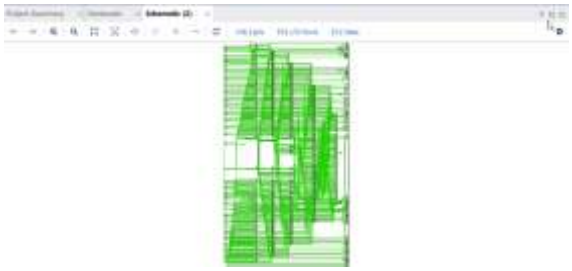

FIGURE: Schematic Diagram

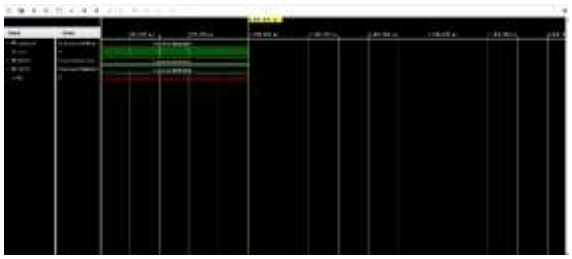

FIGURE: Final Output Of The Project

5. CONCLUSION

In this study, eight algorithms on the serial, parallel, semi-serial, and semi-parallel topologies are mapped to two approximated addition ideas for implementation utilizing memristive IMPLY logic. We demonstrated their advantages by using them in in-memory image processing. To cut down on steps, energy, and space usage, we made use of each adder structure's special benefits. Compared to precise adders, our method saves up to 12% of memristors, 8% to 54% of energy usage, and 6% to 54% of steps. Our method increases the speed by up to 72% and is 9%–43% more energy efficient than existing approximated adders. In terms of both NMED and MRED, we were able to accomplish this while producing a greater accuracy. To the best of our knowledge, we report the first estimated adders in both parallel and semi-parallel topologies, which reduced the process count of an 8-bit addition to just 33 steps. We evaluated the error metrics, confirmed that our adders worked, and incorporated them as

the bottom bits of an RCA. In many real-world image processing applications, we used these partly approximated RCAs. To broaden the scope of our research, we suggested fresh datasets for picture addition and grayscale filtration. With up to 5/8 estimated adders, we were able to obtain satisfactory PSNR measurements of more than 30dB. When compared to SoA approximations, we increased the number of steps by up to 72%, the energy consumption by 9% − 43%, and the image quality by up to 3.4dB of PSNR.

We refined our NoCarry technique to use just one cycle per estimated bit in the image subtraction application. This resulted in a 57% increase in energy efficiency. Additionally, our method produces 3dB–10dB of PSNR more picture quality than SoA estimates. As an illustration of more intricate end uses, we tested our estimated adders on Gaussian smoothing after integrating them into an array multiplier. According to our findings, the suggested approach can approximate up to 86% while still producing results that are satisfactory. Future studies should focus on a more thorough examination of more application-specific approximations and their design.

REFERENCES

[1] V. Gupta, D. Mohapatra, S. P. Park, A. Raghunathan, and K. Roy, "IMPACT: IMPrecise adders for low-power approximate computing," in Proc. IEEE/ACM Int. Symp. Low Power Electron. Design, Aug. 2011, pp. 409–414.

[2] W. Liu, F. Lombardi, and M. Schulte, "A retrospective and prospective view of approximate computing," Proc. IEEE, vol. 108, no. 3, pp. 394–399, Mar. 2020.

[3] R. H. Dennard, J. Cai, and A. Kumar, "A perspective on today's scaling challenges and possible future directions," Solid-State Electron., vol. 51, no. 4, pp. 518–525, 2007.

[4] N. TaheriNejad, "In-memory computing: Global energy consumption, carbon footprint, technology, and products status quo," in Proc. 24th IEEE Int. Conf. Nanotechnol. (IEEE-NANO), 2024, pp. 1–6.

[5] V. Gupta, D. Mohapatra, A. Raghunathan, and K. Roy, "Low-power digital signal processing using approximate adders," IEEE Trans. Comput.-Aided Design Integr. Circuits Syst., vol. 32, no. 1, pp. 124–137, Jan. 2013.

[6] F. Seiler and N. TaheriNejad, "An IMPLY-based semi-serial approximate in-memristor adder," in Proc. IEEE Nordic Circuits Syst. Conf. (NorCAS), Oct. 2023, pp. 1–7.

[7] S. Shakibhamedan, N. Amirafshar, A. S. Baroughi, H. S. Shahhoseini, and N. TaheriNejad, "ACE-CNN: Approximate carry disregard multipliers for energy-efficient CNN-based image classification," IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 71, no. 5, pp. 2280–2293, May 2024.

[8] A. Ibrahim, M. Osta, M. Alameh, M. Saleh, H. Chible, and M. Valle, "Approximate computing methods for embedded machine learning," in Proc. 25th IEEE Int. Conf. Electron., Circuits Syst. (ICECS), Dec. 2018, pp. 845–848.

[9] S. Shakibhamedan, A. Aminifar, L. Vassallo, and N. TaheriNejad, "Harnessing approximate computing for machine learning," in Proc. IEEE Comput. Soc. Annu. Symp., Jun. 2024, pp. 1–6.

[10] H. Jiang, C. Liu, L. Liu, F. Lombardi, and J. Han, "A review, classification, and comparative evaluation of approximate arithmetic circuits," ACM J. Emerg. Technol. Comput. Syst., vol. 13, no. 4, pp. 1–34, Aug. 2017, doi: 10.1145/3094124.

[11] C. Ossimitz and N. TaheriNejad, "A fast line segment detector using approximate computing," in Proc. IEEE Int. Symp. Circuits Syst. (ISCAS), May 2021, pp. 1–5.

[12] H. R. Mahdiani, A. Ahmadi, S. M. Fakhraie, and C. Lucas, "Bio-inspired imprecise computational blocks for efficient VLSI implementation of soft-computing applications," IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 57, no. 4, pp. 850–862, Apr. 2010.

[13] S. E. Fatemieh, S. S. Farahani, and M. R. Reshadinezhad, "LAHAF: Low-power, area-efficient, and high-performance approximate full adder based on static CMOS," Sustain. Comput. Informat. Syst., vol. 30, Jun. 2021, Art. no. 100529.

[14] H. A. Almurib, T. N. Kumar, and F. Lombardi, "Inexact designs for approximate low power addition by cell replacement," in Proc. Design, Autom. Test Eur. Conf. Exhibition, 2016, pp. 660–665.

[15] L. Chua, "Memristor-the missing circuit element," IEEE Trans. Circuit Theory, vols. CT-18, no. 5, pp. 507–519, Sep. 1971.

[16] 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. 6–873, Aug. 2010.

[17] E. Lehtonen and M. Laiho, "Stateful implication logic with memristors," in Proc. IEEE/ACM Int. Symp. Nanoscale Archit., Aug. 2009, pp. 33–36, doi: 10.1109/NANOARCH.2009.5226356.
[18] D. B. Strukov, G. S. Snider, D. R. Stewart, and R. S. Williams, "The missing memristor found," Nature, vol. 453, no. 7191, pp. 80–83, May 2008, doi: 10.1038/nature06932.
[19] A. Karimi and A. Rezai, "Novel design for a memristor-based full adder using a new IMPLY logic approach," J. Comput. Electron., vol. 17, no. 3, pp. 1303–1314, Sep. 2018.
[20] N. TaheriNejad, "SIXOR: Single-cycle in-memristor XOR," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 29, no. 5, pp. 925–935, May 2021.
[21] K. A. Ali, "New design approaches for flexible architectures and in-memory computing based on memristor technologies," Ph.D. thesis, Dept. Electronique, IMT Atlantique-ELEC, Nantes, France, 2020.
[22] D. Radakovits and N. Taherinejad, "BEhavioral leakage and IntEr-cycle variability emulator model for ReRAMs," 2021, arXiv:2103.04179. [23] C. Li et al., "In-memory computing with memristor arrays," in Proc. IEEE Int. Memory Workshop (IMW), May 2018, pp. 1–4.
[24] S. Ganjeheizadeh Rohani, N. Taherinejad, and D. Radakovits, "A semiparallel full-adder in IMPLY logic," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 28, no. 1, pp. 297–301, Jan. 2020.
[25] N. TaheriNejad, T. Delaroche, D. Radakovits, and S. Mirabbasi, "A semi-serial topology for compact and fast IMPLY-based memristive full adders," in Proc. 17th IEEE Int. New Circuits Syst. Conf. (NEWCAS), Jun. 2019, pp. 1–4.
[26] 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.
[27] S. G. Rohani and N. TaheriNejad, "An improved algorithm for IMPLY logic based memristive full-adder," in Proc. IEEE 30th Can. Conf. Electr. Comput. Eng., Apr. 2017, pp. 1–4.
[28] D. Radakovits, N. Taherinejad, M. Cai, T. Delaroche, and S. Mirabbasi, "A memristive multiplier using semi-serial imply-based adder," IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 67, no. 5, pp. 1495–1506, May 2020.
[29] S. E. Fatemieh, M. R. Reshadinezhad, and N. TaheriNejad, "Approximate in-memory computing using memristive IMPLY logic and its application to image processing," in Proc. IEEE Int. Symp. Circuits Syst. (ISCAS), May 2022, pp. 3115–3119.
[30] S. Kvatinsky, A. Kolodny, U. C. Weiser, and E. G. Friedman, "Memristor-based IMPLY logic design procedure," in Proc. IEEE 29th Int. Conf. Comput. Design (ICCD), Oct. 2011, pp. 142–147, doi: 10.1109/ICCD.2011.6081389.
[31] S. Gupta, M. Imani, and T. Rosing, "Felix: Fast and energy-efficient logic in memory," in Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD), Aug. 2018, pp. 1–7.
[32] S. Kvatinsky et al., "MAGIC—Memristor-aided logic," IEEE Trans. Circuits Syst. II, Exp. Briefs, vol. 61, no. 11, pp. 895–899, Nov. 2014.
[33] P. Huang et al., "Reconfigurable nonvolatile logic operations in resistance switching crossbar array for large-scale circuits," Adv. Mater., vol. 28, no. 44, pp. 9758–9764, Nov. 2016, doi: 10.1002/adma.201602418.
[34] S. Asgari, M. R. Reshadinezhad, and S. E. Fatemieh, "Energy-efficient and fast IMPLY-based approximate full adder applying NAND gates for image processing," Comput. Electr. Eng., vol. 113, Jan. 2024, Art. no. 109053.
[35] S. E. Fatemieh, M. R. Reshadinezhad, and N. TaheriNejad, "Fast and compact serial IMPLY-based approximate full adders applied in image processing," IEEE J. Emerg. Sel. Topics Circuits Syst., vol. 13, no. 1, pp. 175–188, Mar. 2023.
[36] 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.
[37] S. Mittal, "A survey of techniques for approximate computing," ACM Comput. Surveys, vol. 48, no. 4, pp. 1–33, Mar. 2016.
[38] H. Jiang, F. J. H. Santiago, H. Mo, L. Liu, and J. Han, "Approximate arithmetic circuits: A survey, characterization, and recent applications," Proc. IEEE, vol. 108, no. 12, pp. 2108–2135, Dec. 2020.
[39] F. Sabetzadeh, M. H. Moaiyeri, and M. Ahmadinejad, "A majority-based imprecise multiplier for ultra-efficient approximate image multiplication," IEEE Trans. Circuits Syst. I, Reg.

Papers, vol. 66, no. 11, pp. 4200–4208, Nov. 2019, doi: 10.1109/TCSI.2019. 2918241.
[40] W. Zhou, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," IEEE Trans. Image Process., vol. 13, no. 4, pp. 600–612, May 2004.
[41] G.-H. Chen, C.-L. Yang, L.-M. Po, and S.-L. Xie, "Edge-based structural similarity for image quality assessment," in IEEE Int. Conf. Acoust. Speed Signal Process. Proc., Jul. 2006, pp. 1–12.
[42] Z. Yang, A. Jain, J. Liang, J. Han, and F. Lombardi, "Approximate XOR/XNOR-based adders for inexact computing," in Proc. 13th IEEE Int. Conf. Nanotechnol., Aug. 2013, pp. 690–693.
[43] S. E. Fatemieh and M. R. Reshadinezhad, "Power-efficient, high-PSNR approximate full adder applied in error-resilient computations based on CNTFETs," in Proc. 20th Int. Symp. Comput. Archit. Digit. Syst. (CADS), Aug. 2020, pp. 1–5.
[44] Y. S. Mehrabani, R. F. Mirzaee, Z. Zareei, and S. M. Daryabari, "A novel high-speed, low-power CNTFET-based inexact full adder cell for image processing application of motion detector," J. Circuits, Syst. Comput., vol. 26, no. 5, May 2017, Art. no. 1750082.
[45] S. Muthulakshmi, C. S. Dash, and S. R. S. Prabaharan, "Memristor augmented approximate adders and subtractors for image processing applications: An approach," Int. J. Electron. Commun., vol. 91, pp. 91–102, Jul. 2018.
[46] S. Muthulakshmi, C. S. Dash, and S. R. S. Prabaharan, "Memristor-based approximate adders for error resilient applications," in Nanoelectronic Materials and Devices. Cham, Switzerland: Springer, 2018, pp. 51–59.
[47] K. Bickerstaff and E. E. Swartzlander, "Memristor-based arithmetic," in Proc. Conf. Rec. Forty 4th Asilomar Conf. Signals, Syst. Comput., Nov. 2010, pp. 1173–1177.
[48] D. Radakovits, M. Jungwirth, S. M. Laube, and N. TaheriNejad. (Sep. 2019). Second (V2.0) LTSpice Implementation of VTEAM. [Online]. Available: https://www.ict.tuwien.ac.at/staff/taherinejad/projects/memristor/files/vteam2.asc
[49] (2017). Knowm. [Online]. Available: https://knowm.org
[50] F. Seiler. (2024). [Online]. Available: https://github.com/fabianseiler/Ax-Image-Processing
[51] A. Prati, I. Mikic, M. M. Trivedi, and R. Cucchiara, "Detecting moving shadows: Algorithms and evaluation," IEEE Trans. Pattern Anal. Mach. Intell., vol. 25, no. 7, pp. 918–923, Jul. 2003.
[52] (2017). USC University of Southern California, Signal and Image Processing Institute (SIPI). [Online]. Available: https://sipi.usc.edu/database/
[53] N. Amirafshar, A. S. Baroughi, H. S. Shahhoseini, and N. TaheriNejad, "Carry disregard approximate multipliers," IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 70, no. 12, pp. 4840–4853, Dec. 2023.
[54] E. Zacharelos, I. Nunziata, G. Saggese, A. G. M. Strollo, and E. Napoli, "Approximate recursive multipliers using low power building blocks," IEEE Trans. Emerg. Topics Comput., vol. 10, no. 3, pp. 1315–1330, Jul. 2022.