ReApprox-PIM: Reconfigurable Approximate Look-Up-Table (LUT)-based Processing-in-Memory (PIM) Machine Learning Accelerator

Sathwika Bavikadi, *Student Member, IEEE,* Purab Ranjan Sutradhar, *Student Member, IEEE,* Mark Indovina, *Member, IEEE,* Amlan Ganguly, *Member, IEEE,* Sai Manoj Pudukotai Dinakarrao, *Member, IEEE*

***Abstract*—Convolutional neural networks (CNNs) have achieved significant success in various applications. Numerous hardware accelerators are introduced to accelerate CNN execution with improved energy efficiency compared to traditional software implementations. Despite the achieved success, deploying traditional hardware accelerators for bulky CNNs on current and emerging smart devices is impeded by limited resources, including memory, power, area, and computational capabilities. Recent works introduced processing-in-memory (PIM), a non-Von-Neumann architecture, which is a promising approach to tackle the problem of data movement between logic and memory blocks. However, as observed from the literature, the existing PIM architectures cannot congregate all the computational operations due to limited programmability and flexibility. Furthermore, the capabilities of the PIM are challenged by the limited available on-chip memory. To enable faster computations and address the limited on-chip memory constraints, this work introduces a novel reconfigurable approximate computing-based PIM, termed ReApprox-PIM. The proposed ReApprox-PIM is capable of addressing the two challenges mentioned above in the following manner: (i) it utilizes a programmable look-up-table (LUT)-based processing architecture that can support different approximate computing techniques via programmability, and (ii) followed by resource-efficient, fast CNN computing via the implementation of highly-optimized approximate computing techniques. This results in improved computing footprint, operational parallelism, and reduced computational latency and power consumption compared to prior PIMs relying on exact computations for CNN inference acceleration at a minimal sacrifice of accuracy. We have evaluated the proposed ReApprox-PIM on various CNN architectures, for inference applications including standard LeNet, AlexNet, ResNet-18, -34, and -50. Our experimental results show that the ReApprox-PIM achieves a speedup of 1.63× with 1.66 × lower area for the processing components compared to the existing PIM architectures. Furthermore, the proposed ReApprox-PIM achieves 2.5× higher energy efficiency and 1.3× higher throughput compared to the state-of-the-art LUT-based PIM architectures.**

S. Bavikadi, S. M. P. Dinakarrao are associated with the Department of Electrical and Computer Engineering, George Mason University, Fairfax, VA, 22030. Email: {sbavikad,spudukot}@gmu.edu

P. R. Sutradhar and A. Ganguly are associated with the Department of Computer Engineering, Rochester Institute of Technology, Rochester, NY, USA. Email: {ps9525,maieee,axgeec}@rit.edu

M. A. Indovina, is associated with the Department of Electrical and Microelectronic Engineering, Rochester Institute of Technology, Rochester USA. Email: maieee@rit.edu

This work was supported in part by the US National Science Foundation (NSF) Grant CNS-2228239. The views, opinions, and/or findings contained in this article are those of the author(s) and should not be interpreted as representing the official views or policies, either expressed or implied, of the US NSF.

***Index Terms*—Processing-in-Memory, Approximate Computing, Convolutional Neural Network, Look-up-Table**

I. Introduction

Recent technological advancements in the field of Machine learning (ML) and Deep learning (DL) to solve complex tasks and are seen to achieve revolutionary outcomes [1]. Convolutional Neural Networks (CNNs) and Deep Neural Networks (DNNs) are widely used ML techniques for applications such as image detection, speech recognition, and various other computer vision-related applications [1]. DNNs and CNNs are data-intensive architectures with billions of hyperparameters, requiring billions of multiply-and-accumulate (MAC) operations [1]. Processing such huge CNN/DNNs in systems with conventional Von Neumann architectures incurs enormous energy consumption and significant execution latency due to the vast data movement between the memory and compute units. To tackle this, non-von Neumann computing paradigms such as In-memory Computing (IMC) *a.k.a.* Processing-in-memory (PIM) have proven to be a potential hardware solution for implementing DNNs and CNNs [2], [3]. PIM architectures alleviate the data movement bottleneck by performing computations locally within the memory. Given that the PIM architectures do not require communication between the main memory and the processing cores, it has proven to improve the data communication efficiency [4], [5].

The PIM paradigms can be classified as Processing using Memory (PuM) architectures that repurpose memory cells/bit-sensing circuitry to implement logic [2], [6] and Processing near Memory (PnM) architectures that incorporate additional processing logic within the memory [7]–[10]. The PuMs execute simple logic across the memory bitline with very high parallel processing bandwidth but are unsuitable for complex tasks [11] as the circuit overheads and the execution latency compound with operational complexity. On the other hand, the PnMs incorporate dedicated computing logic within the memory die for computing but often come with a large processing footprint. This essentially limits the maximum number of parallel PEs and also increases data movement inefficiencies as the datapath becomes lengthier [8].

Performing computations using look-up-tables (LUTs), either in a PuM [3], [12] or PnM [13] layout, offers higher operational flexibility at minimal incremental overheads than the

logic-based computing approach [14]. However, LUT-based computing has its own challenges, including an exponentially increasing footprint with data precision for computations. We observe that this limitation of LUTs, however, can largely be avoided by adopting operation decomposition techniques that allow a large LUT to be replaced with a group of smaller (*i.e.* lower data-precision) LUTs with a notably lower aggregated footprint of the PEs. This acts as a central concept for designing the proposed PE architecture that consists of a group of interconnected, tiny LUTs that can be programmed collectively to execute multiple complex logic/arithmetic operations with larger precisions. The compactness of the PE architecture, in turn, enables more units to be integrated within the memory, resulting in higher operational parallelism.

Additionally, extensive research endeavors in the domain of Approximate Computing (Section II-B) indicate that discarding or simplifying some of the computing workloads results in an improved operational cycle, reduced power consumption, latency, and computing footprint at a minimal sacrifice accuracy at CNN inferences. Further, by developing a programmable in-memory processing architecture, it is possible to facilitate multiple approximate computing techniques for improved CNN inference performances without adding incremental computing overhead for each technique. Specifically, we explore two different forms of approximation computing techniques, each with its own unique advantages and strengths. At the same time, both capitalize on a programmable LUT-based in-memory processing architecture for improved performance, energy, and computing overheads. Our contributions include an efficient implementation of these computing algorithms via a) a reduced precision computing dataflow, and b) computational approximations dataflow, as demonstrated in Section III.

To this end, we introduce a novel DRAM-based reconfigurable approximate PIM (ReApprox-PIM). The proposed ReApprox-PIM utilizes LUTs to perform CNN inferences using multiple approximate computation techniques with the lowest possible operational steps and can be programmed at run time to switch among different approximate computing modes. These computations are distributed across a group of LUTs within the PE, each of which are assigned arithmetic or logical sub-operations, allowing the LUTs to cooperatively implement novel optimized dataflow schemes for executing these computations. We specifically rely on approximate computing techniques that incur minimal CNN inference accuracy losses compared to baseline exact computation techniques while also allow us to achieve significant gains in performance and computational efficiency.

Based on the accuracy of CNN [1] inferences with various bit-precisions of computations on the MNIST dataset shown in Figure 1, the accuracy degradation for downscaling bit-precision from 32-bit to 8-bit is < 2 %, and from 32-bit to 4-bit is < 5 %. However, at the same time, the latter reduces the computing overhead of CNN computations by up to 89%. This leads us to extrapolate 6-bit precision as an ideal midpoint for achieve a balance of inference accuracy and significant performance gains, prompting the implementation of 6-bit precision approximate computation.

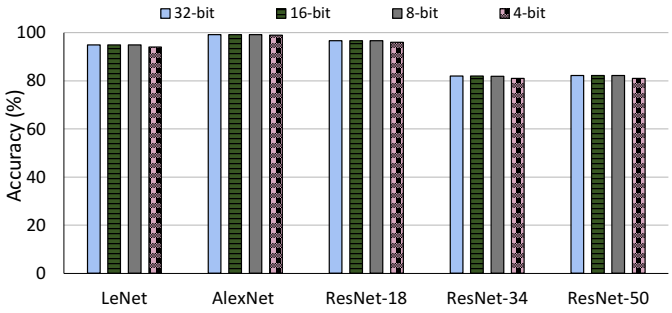

Fig. 1. Top-5 accuracies for different bit precision implemented on LeNet, AlexNet, ResNet-18, 34, and 50

The cardinal contributions of this work could be outlined in a three-fold manner, as presented below:

- We propose a programmable and reconfigurable Look-up-table (LUT)-based processing-in-memory (PIM) architecture that can be reprogrammed in-situ for different quality modes. These unique features make our proposed approximation technique input-adaptive, enabling it to adapt the approximations based on the nature of the inputs and the application.
- We introduce a novel datapath design for the approximate computational strategy for efficient core utilization and faster inference. This also enables a lower memory footprint.
- We evaluate the proposed ReApprox-PIM architecture on various CNN architectures, including LeNet, AlexNet, ResNet-18, -34, and -50, for inference applications and show that it outperforms the state-of-the-art techniques using throughput, energy efficiency, and accuracy.

II. Background and Related Works

We first review some of the existing works on PIM architectures in this section. State-of-the-art approximate computing techniques and their applicability to PIM architectures are discussed in the later part of this section.

A. PIM Architectures

Various PIM architectures have been proposed for ML acceleration [2], [6], [15]. PIM architectures that perform bitwise logical operations are exploited for CNN acceleration and have been built on DRAM (Dynamic Random Access Memory), SRAM (Static Random Access Memory), as well as non-volatile memory technologies such as STT-MRAM (Spin-Transfer Torque Magnetic Random Access Memory) in [16] and MRIMA [17], SOT-MRAM (Spin-Orbit Torque Magnetic Random Access Memory) in [18], and IMCE [19]. The bulk bit-wise PIM architectures have only been able to facilitate fixed-point low data-precision inferences, albeit with overall better performance and efficiency.

Some other works make use of the emerging non-volatile memory elements such as resistive RAM (ReRAM) architectures [4], [20] and leverage crossbar arrays to implement parallel analog computations and are capable of processing floating-point precision data. The analog crossbar arrays are also implemented on other memory platforms such as SRAM-based XNOR-SRAM [21], IMAC [22] and STT-MRAM based

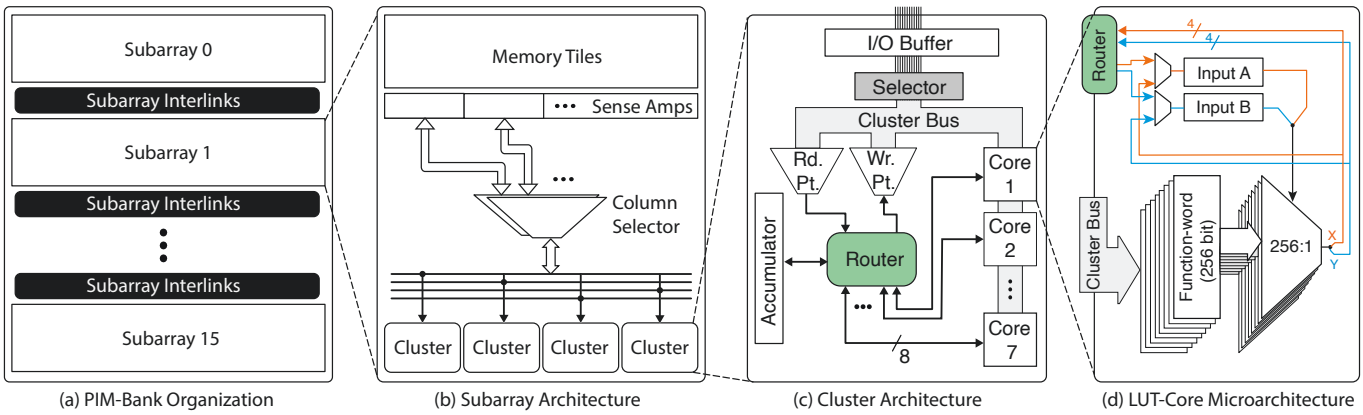

Fig. 2. Overview of the proposed ReApprox-PIM architecture with in-DRAM data-mapping scheme

MRAM-DIMA [23] for CNN acceleration. Despite accelerating the CNNs, they suffer from area and latency overheads due to the analog-to-digital (ADC) and digital-to-analog (DAC) conversions.

Another class of PIM architectures are investigated which uses LUT-based CNN accelerators such as LAcc [3], pluto [12], DLUX [7], BFree [14], pPIM [13] mainly in order to tackle the computational loads of MAC operation. These works leverage modified memory architecture to perform single-stage high-precision matrix multiplications in memory for CNN inferences and achieve high throughput and lower power consumption than the aforementioned PIM architectures.

B. Approximate Computing in PIMs

Several prior works have explored incorporating approximation strategies into memory-centric computing. For example, NeuralPIM [24] utilized neural approximated peripheral circuits in a resistive RAM-based accelerator to minimize energy-intensive ADC/DAC conversions. Tulipa [25], a DRAM-based PIM for image processing applications in particular. Their methodology involves constraining the approximation strategy by setting a lower bound on the Peak Signal-to-Noise Ratio (PSNR) of the processed image frame. Another approach involved generating and mapping approximate reduced ordered binary decision diagrams (ROBDDs) to memristor crossbars for in-memory execution [26] of neural network applications. Various methods [27], [28] were introduced, including system-level and circuit-level approximation through bit trimming and mem-resistance scaling. A hybrid approximate PIM [28] approach allowed dynamic approximation with tunable bit-trimming.

In the realm of non-volatile memory, Pj-AxMTJ [29] employed approximate full adders in MRAM to facilitate efficient magnetization switching in magnetic tunnel junctions. This approach can extend to other bitwise operations, particularly in low-precision computational memory and error-tolerant applications. Another work, ApproxPIM [30], applied approximation techniques from the algorithmic to architectural level in off-the-shelf 3D-stacked DRAM-based PIM. This architecture enabled bitwise logic and substitution operations on tunable precision data, offering potential solutions for memory-wall challenges and energy-efficient system designs. Additionally, another error-resilient approximate hardware accelerators like QLUT [31] focused on accelerating neural networks using approximation MAC units, which may be suitable for computationally expensive applications. This work presents a generic input-aware approximation technique that enables energy-efficient meta-function implementations in error-resilient applications by replacing the constituent complex arithmetic functions with a quantized lookup table (qLUT).

Despite achieving better latency performance, these designs have limited reconfigurability, limiting their applicability across various applications. In contrast, this work introduces a low-latency reconfigurable LUT-based PIM that minimizes data communication overhead and memory bottlenecks. The proposed ReApprox-PIM allows the programming and reprogramming of processing elements (PEs) to handle various computations required for neural network layer operations at different bit widths. Additionally, it incorporates different approximation strategies, offering flexibility in precision and performance. As a result, it supports a broad spectrum of machine learning applications.

III. Proposed ReApprox-PIM Accelerator

The proposed ReApprox-PIM architecture also leverages an approximate multiplier design to substantially enhance the performance. Figure 2 gives the hierarchical architecture view of the ReApprox-PIM with the data mapping scheme, showing the Processing elements in the DRAM chip in (a), LUT cores inside the cluster in (b), router design in (c), and finally the core architecture inside the cluster in (d), This PIM core facilitates logic/arithmetic operations on a pair of data inputs. A router interconnects the cores together to form a cluster, each of these clusters acts as a complete processing element (PE) that can support a wide range of operations such as Multiplication and Accumulation (MAC) to support ML algorithms such as CNNs and DNNs.

A. ReApprox-PIM Architecture Overview

In order to offer a larger degree of functional flexibility, adaptability, and programmability, a LUT-based design is adopted for the PIM core instead of a pre-defined logic

circuit. Moreover, it has been shown in recent works such as [3] and [13], that a LUT-based in-memory arithmetic unit can perform multiplications with a significantly lower delay compared to bitwise computing [2], [6] without any trade-off in the accuracy.

Our proposed ReApprox-PIM consists of processing elements (PEs) called clusters that can support various arithmetic operations, including multiplication, addition, comparison, and encoding multiple bits. These clusters are arranged in rows along the subarrays within DRAM banks, as shown in Figure 2(a), to facilitate data access with low latency. Contrary to off-the-shelf DRAMs, the banks in the proposed architecture are enhanced with subarray interlinks [32] that allow rows of clusters placed along adjacent subarrays to directly communicate at a granularity of one DRAM page at a time. This allows the ReApprox-PIM architecture to easily distribute a particular task among multiple clusters arranged in the columns. At the same time, different columns of the clusters inside the DRAM bank execute parallel and independent tasks in a single instruction multiple data (SIMD) fashion.

Each of these clusters comprises multiple LUT cores as illustrated in Figure 2(b). The LUT cores are interconnected via a crossbar wiring called the router, as shown in Figure 2(c) to facilitate internal communication within the cores and perform complex operations in multiple stages. During an operation, the router establishes parallel communication among the LUT cores and routes the outputs of the cores as inputs for the next operational cycle, if required. This allows the clusters to perform complex operations in multiple steps by establishing direct and parallel communication among the cores.

The LUT cores in the ReApprox-PIM are implemented to perform arithmetic or logical operations on different data width operands with 6-bit precision, on a pair of 3-bit inputs or a single 6-bit input. The LUT-based processing technique relies on pre-computed outputs. A set of six 64-bit 'function-words' are required for reprogramming any LUT.

B. Baseline 8-bit (Exact) computations using LUTs

In this subsection, we present our previously developed LUT-based processing architecture [33]–[40] for supporting 8-bit exact computations using a group of 8-bit LUTs, as shown in Figure 3. To execute this operation, we rely on operation decomposition techniques. The 8-bit multiplicands, A and B are split into 4-bit sections A_H, A_L, B_H, and B_L, respectively. These A and B operands are cross-multiplied in the cores to generate four partial products V_0-V_3:

$$V_0 = A_L * B_L \quad (1)$$

$$V_1 = A_L * B_H \quad (2)$$

$$V_2 = A_H * B_L \quad (3)$$

$$V_3 = A_H * B_H \quad (4)$$

Figure 3(b) shows the routing mechanism of the LUT-cores in a PE to perform multiplication by programming three LUT-cores as 4-bit multipliers and four LUT-cores as 4-bit adders. Finally, the step-wise implementation of the whole process inside a PE is presented in Figure 3(c). The partial products from equation (1)-(4) are added in several stages in order to generate the product of A and B.

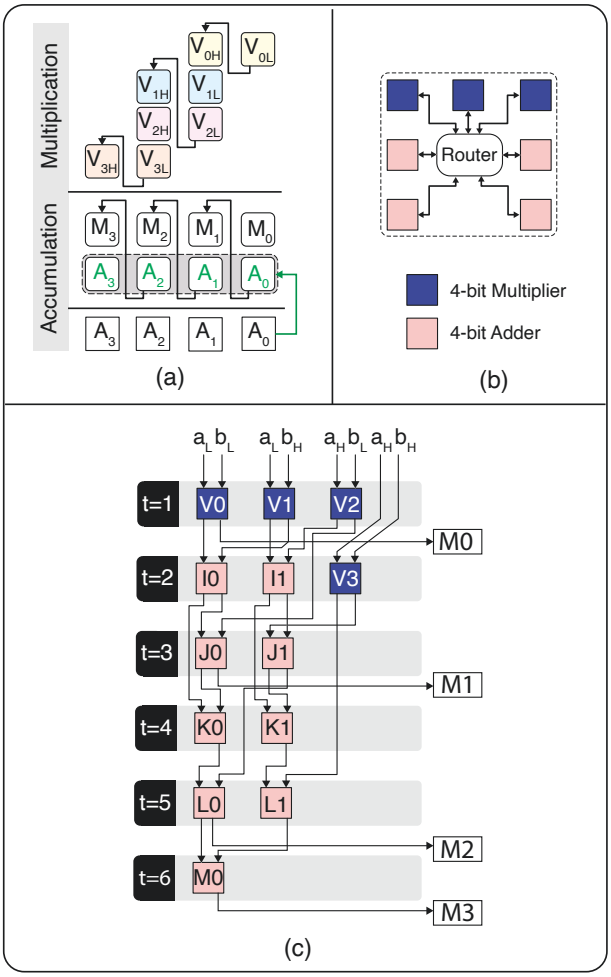

Fig. 3. Overview of 8-bit exact multiplication operation in ReApproxPIM.

C. Novel Approximate Computing on ReApprox-PIM

With the advancement in the ML algorithms, there is an increase in the CNN network size requiring more resources. As a result, the hardware acceleration of ML algorithms is challenging. This necessitates the tuning of the ML algorithms for hardware acceleration in order to attain high performance. However, most modern ML applications can tolerate imprecise computations and do not require maximum accuracy. This realization has motivated the introduction of the Approximate Computing (AC) paradigm, where an inaccurate solution is sufficient in tackling complex problems while enabling high-performance gains for hardware acceleration.

PIM architecture, which modifies the sense amplifier, cannot support multiple processing functionalities simultaneously. Accelerating CNNs or DNNs involves significant data computations, and increasing PIM operations can lead to higher memory costs in terms of area, energy, and overall performance. To address the data movement issue for large data, the proposed ReApprox-PIM architecture utilizes LUTs for addition and multiplication operations in-memory. To further enhance the performance of CNN implementation on the PIM platform, an approximate module is introduced to perform approximate multiplication. This improves the latency and energy efficiency, reduces the overall power consumption and reduces the resource utilization of the convolution operations by significantly reducing the processing workload.

Given that a wide range of weight data and activation maps

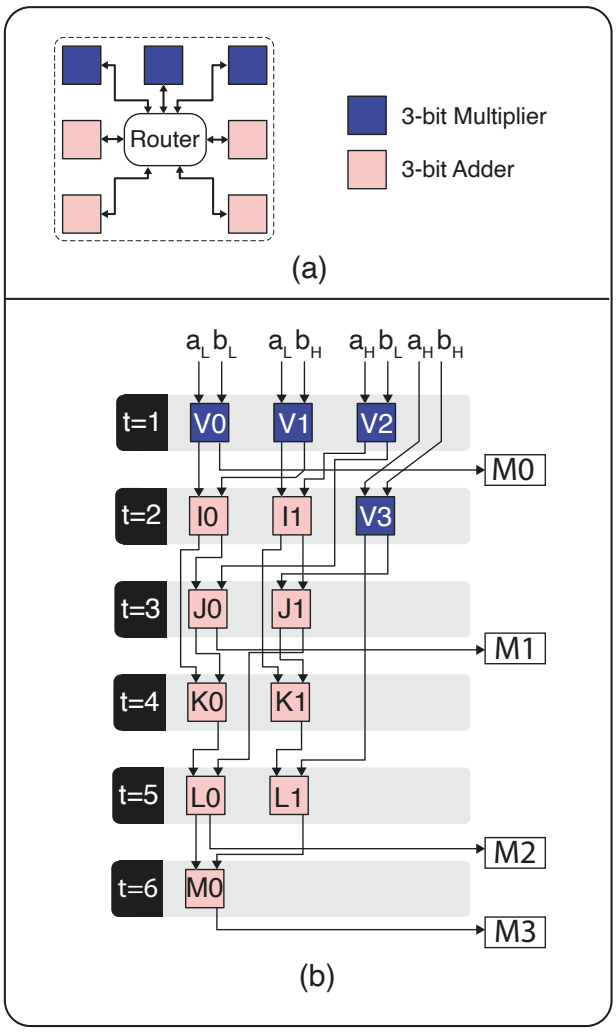

Fig. 4. Overview of Approximate 1 strategy dataflow for multiplication operation in ReApproxPIM.

need to be moved back and forth between PEs and memory units, such data movement represents a key bottleneck in the reduction of CNN's power consumption. Therefore, from an architectural design and datapath perspective, we introduce two approximate computation techniques on PIM to reduce the in-memory computational load: (i) Reduced-Precision Computing datapath, and (ii) Computational Approximation datapath. These approximate computing techniques implemented on PIM architecture are discussed in detail in the next subsections. Specifically, the focus is on MAC operations which are the most frequently used and computationally expensive operations, we demonstrate the use-case where the cluster can be efficiently used in order to perform MAC operations using both approximate computation techniques. We also present the approximation pooling and activation operations in subsection III-D.

1) Approximation 1 Strategy (Reduced Precision Computing datapath): Emerging intelligent applications such as neural networks require a plethora of MAC operations to be performed. Though the proposed PIM architecture supports MAC operations, the amount of memory needed to store the 8-bit data and the computational latency to perform these operations are higher. The data presented in Figure 1 shows that reducing the number of bits i.e., disregarding the LSBs leads to a negligible loss of accuracy when executing CNNs/DNNs [41]. In addition to maintaining accuracy, a reduced-precision approach also reduces the computational steps required for MAC operations thereby reducing the operational latency. Thus, we design a reduced-precision computing-based approximation dataflow for LUT-based PIM in this work. This approximation strategy is outlined in Algorithm 1 where the 6-bit operations are performed by disregarding the 2 LSBs from an 8-bit operand requiring fewer cycles of operations for both read and write operations. Therefore a significant improvement in terms of memory access is observed when the bit width of the LUT cores decreases from 8-bit to 6-bit operations when performing computational operations such as MAC operations.

Algorithm 1: Approximation 1 Strategy

Data: A (3 bit data), B (3-bit data)
Result: Approximate 6-bit multiplier outcome (P)
if $A > 0$, $B > 0$ **then**
  P = $A * B$
end
return P

The step-wise implementation of the proposed datapath for the Approximate algorithm 1 design is shown in Figure 4. This datapath is crafted for the reconfigurable PEs using LUT cores. It divides the operation into distinct modules, involving multipliers and adders, both implemented through the reprogrammability of LUTs.

Initially, the partial products are generated for the input A & B which are passed to the 6-bit multiplier cores. These products are subsequently added at several stages with the help of 6-bit adder cores as shown in Figure 4. Interestingly, implementing a multiplication operation reveals a similar datapath for both exact operation and Approximate 1 operation as demonstrated in Figure 4, Figure 3. Consequently, the same number of cores are required for implementing multiplication operations, whether for an 8-bit or Approximate 1 operation. However, disregarding 2 LSBs in an 8-bit LUT core does not improve the operation datapath nor reduces the number of computational steps as shown in Figure 4. This architectural approach allows sacrifices of accuracy while not sacrificing performance and energy efficiency and is useful for low-power applications. Furthermore, using 8-bit LUT cores for 6-bit data results in performance overheads in terms of area, power, and latency without any improvement in terms of performance. Therefore, in order to further optimize hardware utilization, we replaced the exact operation (8-bit LUT cores) with the Approximate operation (6-bit LUT cores).

While performing an array of multiplication operations on an 8-bit exact operation, the proposed PIM architecture requires seven cores and eight function words containing eight 256-bit latches read by 256-to-1 MUX. On the other hand, replacing the LUT core to perform a 6-bit Approximate 1 operation on the proposed PIM architecture requires seven cores and eight function words containing eight 64-bit latches which are read by 64-to-1 MUX. This significantly improves the area and power utilization by 4×.

2) Approximation 2 Strategy (Computational Approximation datapath: Reducing the bit precision through a hard-

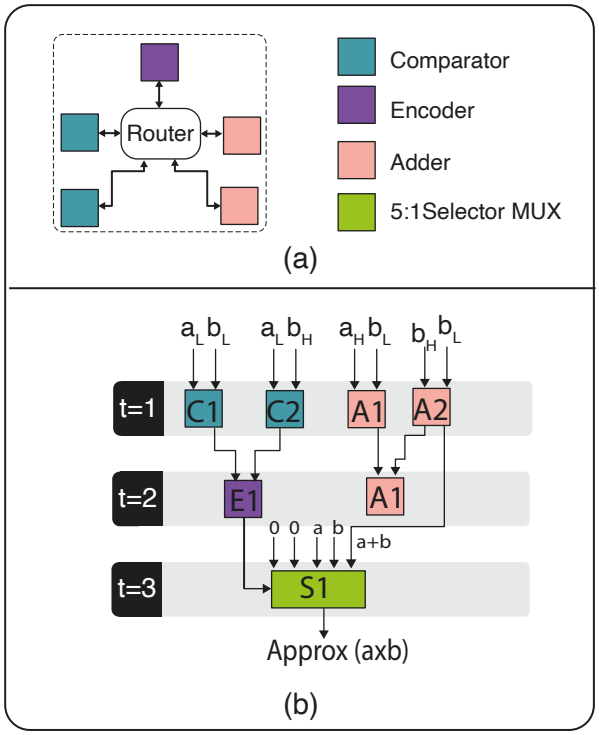

Fig. 5. Overview of Approximate 2 strategy dataflow for multiplication operation in ReApproxPIM.

approximation strategy such as Least Signification Bits (LSBs) truncation leads to low cost for in-memory operations. However, internal data movement is carried out with very low latency when opting for low-precision data, as seen in the Approximation 1 technique. This leads to lower latency and higher energy efficiency. To further improve the speedup and energy-efficiency, and resource utilization of 6-bit LUT cores, we introduce an Approximation 2 technique here. This approximation technique leverages an approximate multiplier design to implement inexact computations and/or approximate computation operations.

To address the issues of delay failures in MSB noted in approximation techniques such as SDC, VOS, ANT [42], [43], we developed simplified approximate units which provide substantial power savings over conventional low-power design techniques. In contrast to these designs which require additional circuitry for implementing approximate array multipliers, leading to area overhead, we have pursued an alternative strategy. To address area overhead concerns and to eliminate the need for specialized additional circuitry, we have introduced a LUT-based approximate design.

CNNs consist of MAC operations that mainly constitute addition and multiplication operations (which in turn, can be built using the adders). Therefore to reduce the computational load in the array architecture, the multiplication is implemented through the addition of partial products generated by multiplying the multiplicand with each bit of multiplier using AND gates. In order to further optimize the computational datapath and resource utilization compared to the Approximation 1 strategy we design a datapath for the Approximation 2 strategy. The goal of the Approximation 2 strategy is to further improve the speed up and energy efficiency of the system. Therefore, for the Approximation 2 strategy, we replace the exact multiplication process with the approximate multiplier as elaborated in Algorithm 2, adopted from the approximate minor adder 5 [44].

Algorithm 2: Approximation 2 Strategy

Data: A (3 bit data), B (3-bit data)
Result: Approximate 2 6-bit multiplier outcome (P)
if $A == 0$ *or* $B == 0$ **then**
  $P = 0$
end
if $A == 1$ **then**
  $P = B$
end
if $A = even$ **then**
  if $B < 8$ **then**
    $P = 0$
  else
    $P = A$
  end
else
  if $B < 8$ **then**
    $P = B$
  else
    $P = B + A$
  end
end
return P

The proposed datapath design of the multiplication process is performed with approximate operations as shown in Figure 5. This datapath is crafted for the reconfigurable PEs using LUT cores by splitting the operation into modular computations, comparisons, encoding, and selections supported by the LUTs using their programmability. The LUT cores programmed to perform addition, comparison, and encoding are used to perform the approximate multiplication operation. In order to perform a 6-bit multiplication operation, both the inputs A&B are split into four 3-bit operands. The set of 3-bit operands A_H, A_L & B_H, B_L are passed to the comparators cores C_1, C_2 respectively, based on the inputs the single-bit comparison output is obtained for each comparison by the 'comparator core'. The results of these two comparators are combined together by the 'encoder core'. The encoder output is used as selection bits to the 5:1 Selector. Based on the selection bits, the 5:1 Selector provides the approximated multiplication outcome. To perform this complete approximate multiplication operation, the proposed PIM architecture requires 5 cores and 3 clock cycles, as shown in Figure 5. Therefore, adapting the approximate multiplication strategy 2 approach on the 6-bit LUT cores increases the speed-up by $2\times$.

D. Approximate Pooling and Activation operations

The pooling layer mainly consists of max-pooling or average pooling operations. These operations can be implemented with a chain of comparison operations. Similarly, Activation operations like Rectified Linear Unit (ReLU) which is, the most dominant filter and computationally efficient activation function, can also be performed by a series of comparison operations. Therefore, both pooling and activation operations are analogous to performing logical AND or OR operations.

For these operations to be carried out efficiently, we utilize LUT cores to store pre-calculated results that facilitate the efficient execution of these logical operations. In the case of pooling, approximation is achieved by comparing values to find the maximum or average. In the case of ReLU activation, approximation is achieved by applying the ReLU function by checking if a value is greater than zero (AND operation) or not (OR operation). In order to perform these two approximate operations, we require 2 LUT cores. Consequently, achieving hardware efficiency in implementing these operations and making them well suited for accelerating neural network computations.

On the other hand, to support activation functions such as Sigmoid, Tanh the proposed ReApprox PIM utilizes the reprogrammable LUTs to implement piecewise linear approximation without requiring additional hardware support as in [45], [46]. These activation functions are implemented as substitution operations where each input value is replaced by the closest 8-bit/6-bit approximation of the output of the desired activation function (Sigmoid, Tanh).

E. Reconfigurable and Input-Aware Approximation

Reconfigurability of the ReApprox architecture enables the approximate mechanisms to support various precision mode operations (8-bit, 6-bit) by modulating the degree of approximation (exact, Approx1, Approx2 mode) required for the CNN application. The key strength of our ReApprox approximation technique is its input-adaptability, enabling it to adapt to the different quality modes of operations such as exact operation or approximate operation based on the nature of the inputs and the specified application. This adaptability is realized by the Input-adaptive Selector, which during the runtime, selects the approximation degree (Approx1, Approx2 mode) as shown in Figure 2. In the proposed system we size the I/O buffer of the Input-adaptive Selector, ensuring that each core can be programmed with the lowest possible programming latency. The latency overhead for reconfiguring the ReApprox PIM cluster is discussed in Section IV-E.

In order to enhance the flexibility and adaptability of the ReApprox architecture, making it better suited for handling a wide range of tasks and efficiently utilizing available resources. We adapt the integration of in-situ reprogramming on the proposed PIM accelerator. In-situ reprogramming allows the proposed ReApprox architecture to switch between different processing modes (exact, Approx1, Approx2) seamlessly. This capability is particularly advantageous for the proposed architecture in scenarios where workloads are diverse and change in real-time. By leveraging in-situ reprogramming, ReApprox PIM can efficiently allocate resources, optimize energy consumption, and simplify implementation, making it a preferred choice for enhancing adaptability to emerging computational requirements.

IV. Evaluation and Results

The proposed architecture is designed to be integrated within the memory banks of a DRAM, which makes it highly modular and adaptable to various DRAM memory configurations such as the Dual Inline Memory Module (DIMM) (DDR4, GDDR6), the 3-D stacked memories (Wide I/O, HMC, HBM/HBM2/HBM2E) all of which share the same bank-level architecture and data management protocol. For simplicity, we have presented the performance evaluation for a configuration with only a single DIMM DRAM chip (DDR4) in the 28nm technology node. However, this design can be scaled up for adoption into larger-scale integration such as 8Hi HBM2E stacks.

In this section, we evaluate the proposed ReApprox-PIM in terms of performance, energy consumption, and area for ML applications. We also compare the ReApprox-PIM with state-of-the-art existing PIM architectures and evaluate them.

A. Overhead comparison analysis

TABLE I
Characteristics of ReApprox-PIM components in 28 nm node

Component	Delay (ns)	Power (mW)	Active Area (μm^2)
ReApprox-PIM (approx1) Core	0.8	2.7	3317
ReApprox-PIM (approx1) Cluster (MAC Operation)	5.6	9.466	23219
ReApprox-PIM (approx2) Core	0.65	0.2924	3317
ReApprox-PIM (approx2) Cluster (MAC Operation)	3.25	0.7875	16585
ReApprox-PIM Core (exact)	0.8	2.7	4196.64
ReApprox-PIM Cluster (MAC Operation for exact computation)	5.6	10.11	29376.48
Intra-Subarray Communication [47]*	63.0	0.028 μJ/comm	N/A
Inter-Subarray Communication [32] for subarrays 1/7/15 hops away*	148.5/ 196.5/ 260.5	0.09/ 0.12/ 0.17 μJ/comm	N/A

**Represented in 28nm technology node*

As a proof of concept, we evaluate AlexNet and implement it on ReApprox-PIM for both the approximation strategies and presented an overhead comparison analysis with the exact computation PIM framework without approximation implementation. We verified the architecture using AISC *via* Verilog HDL implementation. We evaluate the performance using different metrics (*such as* operational latency, power consumption and active area) from HDL synthesis on Synopsys Design Compiler synthesis tools.To make our synthesis compatible with the DRAM technology, we matched the technology node of our Design Compiler synthesis with that of the base DRAM technology node. We also restricted the number of metal layers used in the synthesis to three to make it compliant with the DRAM process technology. The synthesis results are presented in Table I. From Table I, it is observed that the core area for ReApprox PIM with approx 1, approx 2 strategies and PIM core with exact computation is 3317 μm^2, 3317 μm^2, 4196.64 μm^2 respectively. In terms of power, our processing architecture, including the proposed parallel PE count, meets the power rating of existing off-the-shelf memory [48].

Although the approx1 strategy, exact computation implementation has the same dataflow for MAC operation, as discussed in Section III-C, approx1 technique is 1.2 $\times$ area efficient than the exact computation PIM implementation because of the difference in the bit precision of the LUT cores.

From the system-level perspective, the PIM requires 256 PIM clusters in order to perform computational operations for exact 8-bit data precision. Therefore, the estimated area overhead for PIM with exact computation is 7.52 μm^2 across the DRAM chip for placing 256 PIM clusters. Whereas the ReApprox PIM requires 64 clusters and the area overhead across DRAM chip for approx 1, approx 2 strategies are 1.48 μm^2 and 1.06 μm^2, respectively. Therefore, it is observed that adapting the approximation strategy instead of the exact computation of 8-bit fixed point precision significantly improves resource consumption by providing 3.98$\times$ area efficiency.

In the case of an 8-bit LUT, a pair of 4-bit operands are utilized to choose one out of 2^8 (=256) 8-bit pre-computed results of an operation. We store these 256 8-bit entries in eight 256-bit arrays of latches, which we term as eight 256-bit function words in the LUT-core latches, and utilize an 8-bit 256-to-1 multiplexer that allows us to read one entry out of that table. Therefore, each core can be programmed in 8 clock cycles, and the programming latency for the cluster is 64 clock cycles in the case of 8-bit LUTs Similarly, a pair of 3-bit inputs can choose one out of 64 entries in a 6-bit LUT. Therefore, in our Approximate 1 and 2 modes, the function words are sized 64-bit wide, and only 6 of those are required to program the LUT to perform any desired logic/arithmetic operation. Therefore, each core can be programmed in 6 clock cycles, 3 clock cycles, and the programming latency for the cluster is 36 clock cycles, 15 clock cycles in the case of Approximate 1 and 2 modes.

The delay of a single 6-bit or 8-bit MAC performed within a cluster involves computations inside the PIM cores as well as communication among the cores. Due to the optimized data flow for approx 2 strategy, it is observed from the Table I that the ReApprox PIM with approx 2 strategy achieved low power, and low latency when compared to approx 1. The power consumption of the cluster is that of all the cores and the core-to-core communication. The power and delay for intra and inter-subarray data transfers are obtained [32] and [47] due to the low power consumption and the area efficacy observed by the ReApproxPIM architecture. Due to the area efficiency of the ReApprox-PIM, we consider infusing one Re-ApproxPIM bank with 64 PIM clusters per DRAM chip in the entire rank of the DRAM chips for a DIMM (dual in-line memory module).

We also compare the microarchitectural properties of a DRAM-based near-bank accelerator called DLUX [7] that uses LUT-based implementation. The area of a single LUT core in DLUX is 0.0095 mm^2 and a single PE to perform MAC operation is 0.4146 mm^2. It is observed that the proposed ReApprox cores and the cluster area for approx 1, approx 2 strategies are 2.86$\times$, 24.9$\times$, 17.8$\times$ respectively. The DLUX used a 1:1 ratio for bank-to-PE and energy-wise, each PE of DLUX costs 279.8pJ/access. In comparison, the baseline operation of ReApprox PIM PE with 8-bit LUT and 1:64 ratio for bank-to-PE costs only 0.084 J/access. Although DLUX performs LUT-based computations for DL acceleration, there is a significant difference in the architectures. DLUX is programmed to perform full-precision floating point operations for high-performance acceleration and is mainly used for training purposes. Whereas the proposed 2-D DRAM-based architecture performs convolutional operations on low-precision fixed operations to optimize the performance.

B. Accuracy Comparison

We evaluate the accuracies for KNN, K-means algorithms along with various CNN models such as LeNet, AlexNet, and ResNet-18,34,50 on the MNIST dataset (28x28x1 image dimensions) and CIFAR-10 dataset (32x32x3 image dimensions). Both datasets consist of about 60,000 training and 10,000 testing images belonging to 10 classes. The goal is to classify the given input image into the correct class. The baseline accuracy of the networks with the 8-bit precision data in comparison with exact, approx1, approx 2 strategies are as shown in Figure 6(a), and Figure 6(b), respectively. The accuracies for KNN, K-means algorithms with exact, approx1, approx2 are 95.2%, 92%, 92.10% and 89.6%, 86.16%, 86% respectively on the MNIST dataset and 85.5%, 82%, 82% and 82.2%, 80.1%, 79.8% for CIFAR-10 dataset, as demonstrated in Figure 6(a), and Figure 6(b), respectively. Similarly, the accuracies for AlexNet with exact, approx1, approx2 are 99.17%, 99.11%, 99.10% respectively on the MNIST dataset and 82.57%, 82.52%, 82.52% for CIFAR-10 dataset is observed from the Figures 6(a), and 6(b), respectively. Although it is observed that inference of the CIFAR-10 dataset is lower than the MNIST dataset it is also observed that the accuracies of both approximation strategies are similar to the exact computation inference for both datasets. The performance degradation between the exact computation, approx 1 strategy is around 0.02%-0.05% for all the CNNs deployed. Similarly, the performance difference between the approx 1 and approx 2 strategies is between 0.01%-0.03%. Therefore, Figure 6(a), and Figure 6(b) suggests that adapting ReApprox-PIM for low-precision application can guarantee good performance results in terms of accuracy.

Approximation strategies implemented on the FPGA-based accelerators such as MBM and [49] SIMDive [50] for ML applications have shown significant results. MBM [49], and SIMDive [50] have implemented 3-layered ANN consisting of three fully connected layers on the MNIST dataset with 8-bit fixed point precision and achieved an accuracy of 96.22%, 96.17% respectively. From Figure 6 (a), it is observed that ReApproxPIM is capable of accelerating deep neural networks ranging from LeNet with 5 layers, AlexNet with 8 layers to ResNet-50 with 50 layered and achieved an accuracy of 98.80%, 99.1%, and 92.42% on the MNIST dataset with 6-bit approximated data. Therefore, considering approximate computing accelerators for ML applications such as [49], [50] ReApproxPIM is capable of implementing image classification tasks efficiently.

Since other approximate PIM approaches [27], [28], [30] are capable of deploying ML algorithms such as KNN, K-means whereas the ReApprox-PIM is capable of deploying more compute-intensive networks such as CNN/DNNs. Therefore, it is also evident that ReApproxPIM is more efficient for data-intensive machine learning applications like image classification.

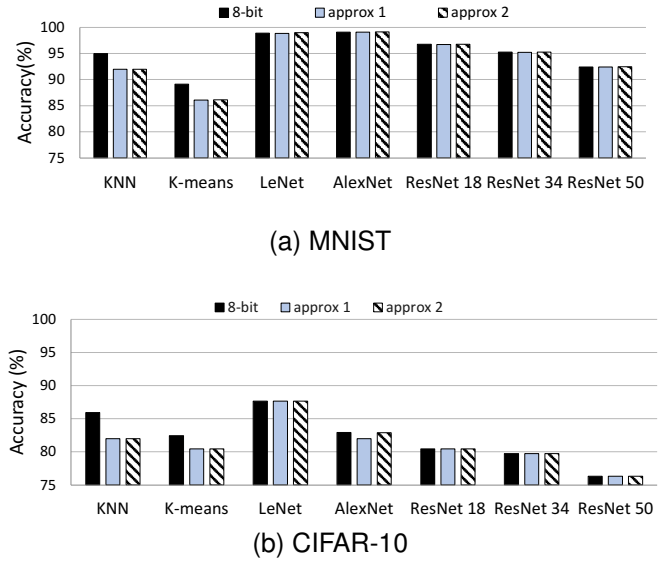

Fig. 6. Accuracy comparison of LeNet, AlexNet, Resnet-18, -34, -50 on MNIST and CIFAR-10 datasets for exact, approx1, approx2 strategies.

C. Efficiency Evaluation

In this section, we present the performance of ReApprox-PIM for these ML algorithms in terms of processing latency (s), energy consumption per frame of an image (Joules/frame), and energy density product (Joules second). Performance of the ReApprox-PIM is evaluated on CNNs when implemented for various inputs such as exact 8-bit, and 6-bit precision with approx 1, approx 2 strategies. The benchmark algorithms are LeNet, AlexNet, ResNet-18, -34, and -50. The ReApprox-PIM requires 64 clusters to implement approx 1, approx 2 strategies, for 8-bit implementation 256 PIM clusters are required. As mentioned in Section IV-A, due to the low power consumption and the area efficacy observed in the ReApprox PIM we consider using a ReApprox PIM bank with 64 PIM clusters per DRAM chip in the complete rank of DRAM chips for a DIMM. Therefore, for evaluation purposes in this section, we consider one ReApprox PIM implementation for a DIMM.

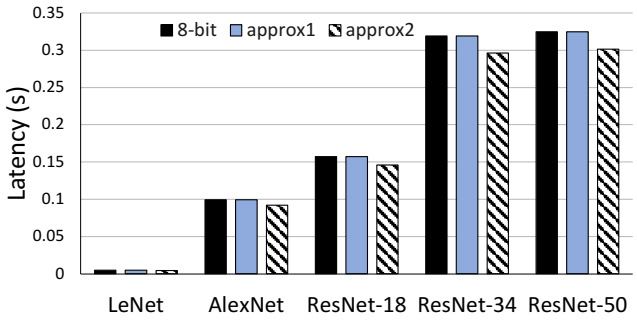

Fig. 7. Performance evaluation in terms of latency of the CNNs implemented on the ReApproxPIM with exact, approx 1, approx 2 strategies

Figure 7 shows the performance comparison plot in terms of latency for the CNNs implemented on ReApproxPIM architecture with input data of exact 8-bit, 6-bit precision with approx 1, approx 2 strategies. When deploying the CNNs with the 6-bit approx 1 input and 8-bit input requires almost the same amount of time to implement on ReApproxPIM and the approx 2 strategy performs at relatively lower latency as demonstrated in Figure 7.

Figure 8 (a) shows the performance comparison plot of the CNNs implemented on ReApproxPIM architecture with approx 1, approx 2 and exact computation strategies in terms of energy efficiency. Similarly, for the same input Figure 8(b) shows the energy-delay product plot of CNNs deployed on the ReApprox PIM. The PIM architecture with exact computation (8-bit precision) requires larger computations than approximate computation operations. Therefore, exact computation implementation requires more energy consumption than approximate strategy implementations as shown in Figure 8 (a), Figure 8(b). As discussed in Section III, the approx 2 strategy requires a few number of cores, and a few operational cycles to implement computational operations compared to approx 1 strategy. Therefore, it is also observed that the approx2 strategy is a more energy-efficient approach when compared to the approx1 approach.

It is observed from Figure 8 (a), (b), Figure 7 that the ReApprox-PIM is capable of performing at low latency and high energy efficiency. Figure 7 shows that it requires a few milliseconds to process the computations required for any CNN. For example, the ResNet-50 with 50 layers and 26 billion computations is processed under 10 ms. Similarly, in Figure 8(a), (b), high energy efficiency is observed for 6-bit approx 1, approx 2 strategies compared to 8-bit implementation due to the lower computations adopted by approximation strategies. Considering other approximate PIM architectures [30], that requires about a second to two of execution time for implementing K-means or KNN algorithm, whereas ReApprox-PIM can implement deep neural networks at impressively low latency.

D. Comparative Performance Evaluation with State-of-the-Art

Based on data garnered from published literature [3], [13], [15], we present a comparative analysis of ReApprox-PIM with four other state-of-the-art PIM architectures: Neural PIM [24], DRISA [2], LACC [3], and pPIM [13] in Figure 9 in terms of area (mm^2) and energy efficiency per unit area (Frames/Joule/mm^2) for AlexNet inference with a batch size of 64. The evaluations of DRISA and LACC are scaled to 8-bit data implementation. All the PIMs in comparison are based on a DRAM platform.

The core area for the ReApprox-PIM with approx 2 strategy obtained from 28nm technology as mentioned in Table I is 3317 μm^2, which is more area-efficient than the approx 1 strategy. Therefore, we consider ReApprox-PIM with approx2 strategy for the comparative analysis in this Section. Since the memory cells in the PIM do not participate in the computing process, we compute the area efficiency using the aggregated area of the PIM clusters. This provides the area overhead of 1.06 mm^2 for 64 ReApproxPIM clusters across the DRAM.

From the Figure 9, the increased area efficiency observed in LACC and pPIM with respect to DRISA is due to the LUT-based multipliers. LACC reserves 4K lines per bank (of size 16K) for the LUTs and DRISA occupies roughly 40% chip area for implementing the logic, leading to inefficient memory space utilization. The LUT-based PIMs [3], [13], in general,

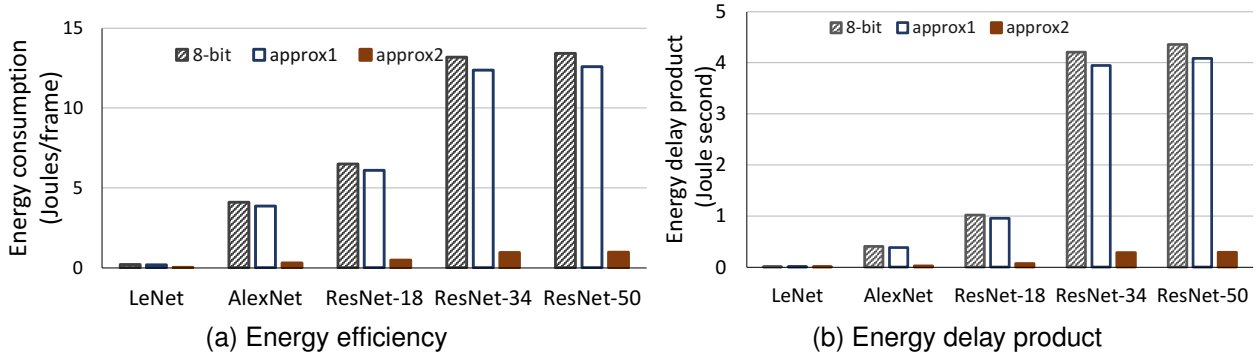

Fig. 8. Performance evaluation in Energy efficiency and Energy delay product of the CNNs implemented on the ReApproxPIM with approx1, approx1 and exact computation strategies.

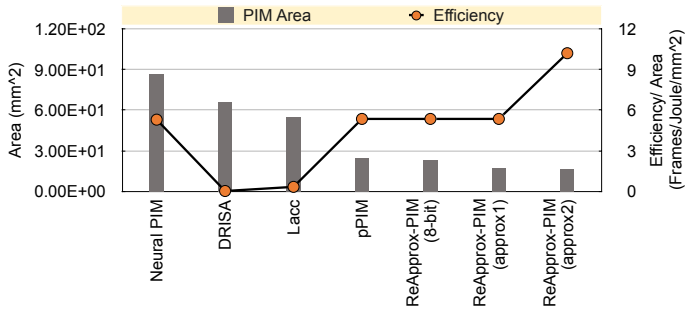

Fig. 9. Comparative performance analysis in terms of Area and Efficiency/Area of ReApproxPIM with respect to state-of-the-art PIM architectures

have higher energy efficiency per unit area as they leverage memory to implement their functionalities.

On the other hand,in comparison to other approximate PIM designs such as Neural PIM [24] with an active area of 86.4 mm^2 and an area-efficiency of 34 μJ/ mm^2, ReApprox-PIM achieves almost 1.8 mJ/mm^2 efficiency per area with an active area of only 16.06 mm^2.

When performing bit trimming operations for CNN application hybrid approximation, PIM accelerators such as [27], [28] have energy consumption of 4.06 J, 73.31 J and speed up as 2.74s, 3.91s for SRAM-based processor, ReRAM based processor respectively. Whereas ReApprox-PIM achieves 4.65 mJ energy efficiency and latency of 0.092 s for AlexNet implementation.

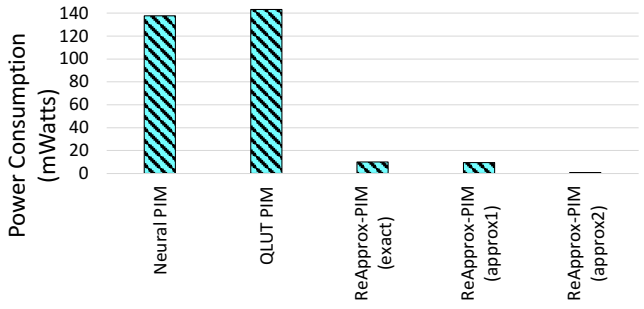

Fig. 10. Comparative performance analysis in terms of Power Consumption of ReApproxPIM with respect to previous Approximate Computing based PIM architectures

Figure 10 shows a performance comparison of Power Consumption (mWatts) for ReApprox-PIM and other Approximate Computing based PIM accelerators such as Neural PIM [24], QLUT [31]. It is observed that the ReApprox-PIM had 14× and 13× better performance in terms of power consumption than Neural PIM and QLUT, respectively.

Figure 11 shows a performance comparison of throughput (frames/second) for ReApprox-PIM and other PIM accelerators. The PIM architectures under comparison include DRAM-based bulk bit-wise processing devices DRISA [2], and DrAcc [6], SRAM-implemented Neural Cache [51], another LUT-based PIM implemented on the DRAM platforms such as LAcc [3], and pPIM architecture [13].

Neural Cache [51] is the slowest among the PIMs studied here due to its smaller processing capabilities as well as its comparatively slower bit-serial computing mechanism. On the other hand, a relatively higher throughput is observed for DRISA [2] due to its ability to parallelize operations across multiple banks. Whereas DrAcc [6] implements 8-bit ternary precision inferences through very minimal circuit modifications which allows it to obtain high performance similar to that of pPIM [13]. The benefits of adopting LUTs in order to utilize pre-calculated results instead of performing in-memory logic operations are convincingly demonstrated by LAcc [3] which achieves impressive inference performances. ReApprox PIM in both the approximation modes has higher AlexNet throughput than all the other PIMs under comparison. The LUTs in the ReApprox-PIM are capable of reprogramming at run-time to perform approximate computational operations and provide flexibility on adapting to different bit-widths, whereas the PIM architectures such as DRISA are not run-time programmable. It is also observed that ReApprox-PIM with approx 2 strategy outperforms the DRISA by 1.26 × and LAcc and pPIM by almost 2.1 × for AlexNet inference.

E. Overhead of Reprograming the ReApprox PIM Clusters

In the proposed system for exact 8-bit operation, we require $2^{2n} \times 2n$ function words. The I/O buffer is designed with a width of 2^{2n} bits and connected to a 2^{2n} bit width bus. This configuration allows each LUT core to be reconfigured and programmed in only $2 * n$ steps through $2 * n$ function writes.

In this approach, the proposed Cluster requires to access memory once for re-writing each function-word, the program-

TABLE II
OVERHEAD OF RECONFIGURABLE PIM CLUSTERS IN THE REAPPROX PIM

	Exact (8-bit) to Approx1	Exact (8-bit) to Approx2	Approx 1 to Exact(8-bit)	Approx 1 to Approx2	Approx 2 to Exact(8-bit)	Approx 2 to Approx1
Latency	2906.88 ns	2849.28 ns	17441.28 ns	2849.28 ns	17441.28 ns	2906.88 ns

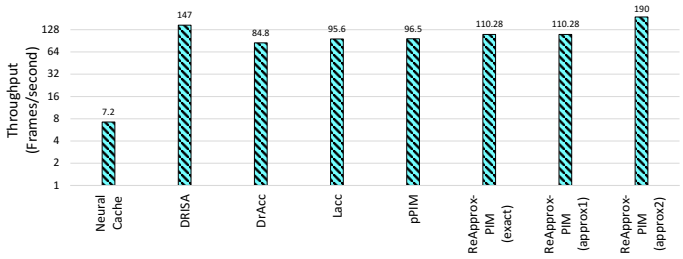

Fig. 11. Comparative performance analysis in terms of Throughput of ReApproxPIM with respect to state-of-the-art PIM architectures

ming latency is dependent on the memory access latency, as well as, the number of memory accesses. Since the data are directly accessed from the subarray's local row buffer, the data access latency is same as the tRCD [52] value of DRAM (*e.g.* 6.77 ns for DDR4 DRAM). Additionally, rewriting a function word in a core can be performed in a single clock cycle, thanks to the internal bus of the same width as the buffer. This gives us an estimated optimum programming latency of a Cluster, which is reported in the Table II.

V. CONCLUSION

In this paper, we introduce reconfigurable approximate computing-based processing in-memory architecture called ReApprox PIM, consisting of look-up-tables to perform in-memory computations. The LUTs in the PIM are capable of reprogramming the processing elements to support multiple precision with both approximate and exact computation during the run-time. The proposed LUT-based ReApprox-PIM provides the ability to reconfigure its functionality to any arbitrary operation. Therefore it is capable of performing computational and logical operations required for convolutional, pooling, activation, and fully-connected layers for CNN inference applications. ReApprox-PIM leverages two types of approximation strategies (i) Reduced-Precision Computing, and (ii) Computational Approximation, to reduce the computational load and enable a low memory footprint. ReApprox-PIM allows sacrificing of accuracy without allowing any performance degradation in terms of energy efficiency, and throughput which makes it ideal for low-power applications. The major benefit of the proposed ReApprox-PIM reducing data movement and lowering storage needs for bulky DNNs and CNNs. Performance evaluation and comparison with respect to state-of-the-art GPU, and other PIM architectures, as well as across operating modes with different bit-precisions of inputs and weights. ReApprox-PIM in both the approximation modes has higher throughput, and energy efficiency when compared to other state-of-the-art PIMs. Our experimental results show that the ReApprox-PIM achieves a speedup of 1.63× with 1.66 × lower area for the processing components compared to the existing PIM architectures. Furthermore, the experiments also show that compared to recent LUT-based PIM accelerators ReApprox-PIM can gain a speedup of 1.3× and energy-efficiency of 2.5× while maintaining the comparable trade-off in terms of accuracy.

REFERENCES

[1] A. A. Awan, H. Subramoni, and D. K. Panda, "An in-depth performance characterization of cpu- and gpu-based dnn training on modern architectures," in *Proceedings of the Machine Learning on HPC Environments*, ser. MLHPC'17. New York, NY, USA: Association for Computing Machinery, 2017. [Online]. Available: https://doi.org/10.1145/3146347.3146356

[2] S. L. et al., "DRISA: A DRAM-based reconfigurable in-situ accelerator," in *IEEE/ACM Int. Symp. on Microarchitecture (MICRO)*, 2017.

[3] Q. Deng and et al., "Lacc: Exploiting lookup table-based fast and accurate vector multiplication in DRAM-based CNN accelerator," in *ACM/IEEE Design Automation Conf. (DAC)*, 2019.

[4] S. Bavikadi, P. R. Sutradhar, K. N. Khasawneh, A. Ganguly, and S. M. Pudukotai Dinakarrao, "A review of in-memory computing architectures for machine learning applications," in *GLSVLSI*, 2020.

[5] S. Bavikadi, A. Dhavlle, A. Ganguly, A. Haridass, H. Hendy, C. Merkel, V. J. Reddi, P. R. Sutradhar, A. Joseph, and S. M. Pudukotai Dinakarrao, "A survey on machine learning accelerators and evolutionary hardware platforms," *IEEE Design & Test*, vol. 39, no. 3, pp. 91–116, 2022.

[6] Q. D. et al., "DrAcc: A DRAM based accelerator for accurate CNN inference," in *ACM/IEEE Design Automation Conf. (DAC)*, 2018.

[7] P. Gu, X. Xie, S. Li, D. Niu, H. Zheng, K. T. Malladi, and Y. Xie, "Dlux: A lut-based near-bank accelerator for data center deep learning training workloads," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 40, no. 8, pp. 1586–1599, 2021.

[8] D. I. Cutress, "Hot chips 31 analysis: In-memory processing by upmem," Aug 2019. [Online]. Available: https://www.anandtech.com/show/14750/hot-chips-31-analysis-inmemory-processing-by-upmem

[9] Y.-C. Kwon, S. H. Lee, J. Lee, S.-H. Kwon, J. M. Ryu, J.-P. Son, O. Seongil, H.-S. Yu, H. Lee, S. Y. Kim, Y. Cho, J. G. Kim, J. Choi, H.-S. Shin, J. Kim, B. Phuah, H. Kim, M. J. Song, A. Choi, D. Kim, S. Kim, E.-B. Kim, D. Wang, S. Kang, Y. Ro, S. Seo, J. Song, J. Youn, K. Sohn, and N. S. Kim, "25.4 a 20nm 6gb function-in-memory dram, based on hbm2 with a 1.2tflops programmable computing unit using bank-level parallelism, for machine learning applications," in *2021 IEEE International Solid- State Circuits Conference (ISSCC)*, vol. 64, 2021, pp. 350–352.

[10] Y. Kwon, K. Vladimir, N. Kim, W. Shin, J. Won, M. Lee, H. Joo, H. Choi, G. Kim, B. An, J. Kim, J. Lee, I. Kim, J. Park, C. Park, Y. Song, B. Yang, H. Lee, S. Kim, D. Kwon, S. Lee, K. Kim, S. Oh, J. Park, G. Hong, D. Ka, K. Hwang, J. Park, K. Kang, J. Kim, J. Jeon, M. Lee, M. Shin, M. Shin, J. Cha, C. Jung, K. Chang, C. Jeong, E. Lim, I. Park, J. Chun, and S. Hynix, "System architecture and software stack for gddr6-aim," in *2022 IEEE Hot Chips 34 Symposium (HCS)*, 2022, pp. 1–25.

[11] C. Sudarshan, M. H. Sadi, L. Steiner, C. Weis, and N. Wehn, "A critical assessment of dram-pim architectures - trends, challenges and solutions," in *Embedded Computer Systems: Architectures, Modeling, and Simulation*, A. Orailoglu, M. Reichenbach, and M. Jung, Eds. Cham: Springer International Publishing, 2022, pp. 362–379.

[12] J. D. Ferreira, G. Falcao, J. Gomez-Luna, M. Alser, L. Orosa, M. Sadrosadati, J. S. Kim, G. F. Oliveira, T. Shahroodi, A. Nori, and O. Mutlu, "pLUTo: Enabling massively parallel computation in DRAM via lookup tables," in *2022 55th IEEE/ACM International Symposium on Microarchitecture (MICRO)*. IEEE, oct 2022. [Online]. Available: https://doi.org/10.1109%2Fmicro56248.2022.00067

[13] P. R. Sutradhar, M. Connolly, S. Bavikadi, S. M. Pudukotai Dinakarrao, M. A. Indovina, and A. Ganguly, "pPIM: A programmable processor-in-memory architecture with precision-scaling for deep learning," *IEEE Computer Architecture Letters*, vol. 19, no. 2, pp. 118–121, 2020.

[14] A. K. Ramanathan, G. S. Kalsi, S. Srinivasa, T. M. Chandran, K. R. Pillai, O. J. Omer, V. Narayanan, and S. Subramoney, "Look-up table based energy efficient processing in cache support for neural network acceleration," in *2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2020, pp. 88–101.

[15] S. M. Shivanandamurthy, I. G. Thakkar, and S. A. Salehi, "Atria: A bit-parallel stochastic arithmetic based accelerator for in-dram cnn processing," in *2021 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, 2021, pp. 200–205.

[16] Y. Pan and et al., "A multilevel cell stt-mram-based computing in-memory accelerator for binary convolutional neural network," *IEEE Transactions on Magnetics*, 2018.

[17] S. Angizi and et al., "Mrima: An mram-based in-memory accelerator," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2020.

[18] D. Fan and S. Angizi, "Energy efficient in-memory binary deep neural network accelerator with dual-mode sot-mram," in *IEEE International Conference on Computer Design (ICCD)*, 2017.

[19] S. A. et al., "IMCE: Energy-efficient bit-wise in-memory convolution engine for deep neural network," in *Asia and South Pacific Design Automation Conf. (ASP-DAC)*, 2018.

[20] P. C. et al., "Prime: A novel processing-in-memory architecture for neural network computation in reram-based main memory," in *ACM/IEEE International Symposium on Computer Architecture (ISCA)*, 2016.

[21] S. Y. et al., "Xnor-sram: In-memory computing sram macro for binary/ternary deep neural networks," *IEEE Journal of Solid-State Circuits*, 2020.

[22] A. M. et al., "Imac: In-memory multi-bit multiplication and accumulation in 6t sram array," *IEEE Transactions on Circuits and Systems I: Regular Papers*, 2020.

[23] A. D. P. et al., "An mram-based deep in-memory architecture for deep neural networks," in *IEEE International Symposium on Circuits and Systems*, 2019.

[24] W. Cao, Y. Zhao, A. Boloor, Y. Han, X. Zhang, and L. Jiang, "Neural-pim: Efficient processing-in-memory with neural approximation of peripherals," *IEEE Transactions on Computers*, pp. 1–1, 2021.

[25] A. Dube, A. Wagle, G. Singh, and S. Vrudhula, "Tunable precision control for approximate image filtering in an in-memory architecture with embedded neurons," in *Proceedings of the 41st IEEE/ACM International Conference on Computer-Aided Design*, ser. ICCAD '22. New York, NY, USA: Association for Computing Machinery, 2022. [Online]. Available: https://doi.org/10.1145/3508352.3549385

[26] A. U. Hassen and S. Anwar Khokhar, "Approximate in-memory computing on reram crossbars," in *2019 IEEE 62nd International Midwest Symposium on Circuits and Systems (MWSCAS)*, 2019, pp. 1183–1186.

[27] H. E. Yantir, A. M. Eltawil, and F. J. Kurdahi, "Approximate memristive in-memory computing," *ACM Trans. Embed. Comput. Syst.*, vol. 16, no. 5s, Sep. 2017. [Online]. Available: https://doi.org/10.1145/3126526

[28] H. E. Yantır, A. M. Eltawil, and F. J. Kurdahi, "A hybrid approximate computing approach for associative in-memory processors," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 8, no. 4, pp. 758–769, 2018.

[29] H. Cai, H. Jiang, M. Han, Z. Wang, Y. Wang, J. Yang, J. Han, L. Liu, and W. Zhao, "Pj-axmtj: Process-in-memory with joint magnetization switching for approximate computing in magnetic tunnel junction," in *2019 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, 2019, pp. 111–115.

[30] Y. Tang, Y. Wang, H. Li, and X. Li, "Approxpim: Exploiting realistic 3d-stacked dram for energy-efficient processing in-memory," in *2017 22nd Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2017, pp. 396–401.

[31] A. Raha and V. Raghunathan, "Qlut: Input-aware quantized table lookup for energy-efficient approximate accelerators," *ACM Trans. Embed. Comput. Syst.*, vol. 16, no. 5s, sep 2017. [Online]. Available: https://doi.org/10.1145/3126531

[32] K. K. Chang, P. J. Nair, D. Lee, S. Ghose, M. K. Qureshi, and O. Mutlu, "Low-cost inter-linked subarrays (lisa): Enabling fast inter-subarray data movement in dram," in *IEEE Int. Symp. on High Performance Computer Arch (HPCA)*, March 2016, pp. 568–580.

[33] P. R. Sutradhar, S. Bavikadi, M. Connolly, S. K. Prajapati, M. A. Indovina, S. M. Pudukotaidinakarrao, and A. Ganguly, "Look-up-table based processing-in-memoryarchitecture with programmable precision-scalingfor deep learning applications," *IEEE TPDS*, 2021.

[34] S. Bavikadi, P. R. Sutradhar, A. Ganguly, and S. M. P. Dinakarrao, "Heterogeneous multi-functional look-up-table-based processing-in-memory architecture for deep learning acceleration," in *2023 24th International Symposium on Quality Electronic Design (ISQED)*, 2023, pp. 1–8.

[35] S. Bavikadi, P. R. Sutradhar, M. Indovina, A. Ganguly, and S. M. P. Dinakarrao, "Reconfigurable processing-in-memory architecture for data intensive applications," in *2024 37th International Conference on VLSI Design and 2024 23rd International Conference on Embedded Systems (VLSID)*, 2024, pp. 222–227.

[36] S. Kasarapu, S. Bavikadi, and S. M. P. Dinakarrao, "Processing-in-memory architecture with precision-scaling for malware detection," in *2024 37th International Conference on VLSI Design and 2024 23rd International Conference on Embedded Systems (VLSID)*, 2024, pp. 529–534.

[37] S. Bavikadi, P. R. Sutradhar, M. A. Indovina, A. Ganguly, and S. M. P. Dinakarrao, "Polar: Performance-aware on-device learning capable programmable processing-in-memory architecture for low-power ml applications," in *2022 25th Euromicro Conference on Digital System Design (DSD)*, 2022, pp. 889–898.

[38] P. R. Sutradhar, S. Bavikadi, S. M. P. Dinakarrao, M. A. Indovina, and A. Ganguly, "3dl-pim: A look-up table oriented programmable processing in memory architecture based on the 3-d stacked memory for data-intensive applications," *IEEE Transactions on Emerging Topics in Computing*, pp. 1–13, 2023.

[39] S. Bavikadi, P. R. Sutradhar, A. Ganguly, and S. M. P. Dinakarrao, "upim: Performance-aware online learning capable processing-in-memory," in *2021 IEEE 3rd International Conference on Artificial Intelligence Circuits and Systems (AICAS)*, 2021, pp. 1–4.

[40] P. R. Sutradhar, S. Bavikadi, M. Indovina, S. M. Pudukotai Dinakarrao, and A. Ganguly, "Flutpim: A look-up table-based processing in memory architecture with floating-point computation support for deep learning applications," in *Proceedings of the Great Lakes Symposium on VLSI 2023*, ser. GLSVLSI '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 207–211. [Online]. Available: https://doi.org/10.1145/3583781.3590313

[41] T. Liang, J. Glossner, L. Wang, S. Shi, and X. Zhang, "Pruning and quantization for deep neural network acceleration: A survey," 2021.

[42] G. V. Varatkar and N. R. Shanbhag, "Error-resilient motion estimation architecture," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 16, no. 10, p. 1399–1412, oct 2008. [Online]. Available: https://doi.org/10.1109/TVLSI.2008.2000675

[43] D. Mohapatra, G. Karakonstantis, and K. Roy, "Significance driven computation: A voltage-scalable, variation-aware, quality-tuning motion estimator," in *Proceedings of the 2009 ACM/IEEE International Symposium on Low Power Electronics and Design*, ser. ISLPED '09. New York, NY, USA: Association for Computing Machinery, 2009, p. 195–200. [Online]. Available: https://doi.org/10.1145/1594233.1594282

[44] V. Gupta, D. Mohapatra, A. Raghunathan, and K. Roy, "Low-power digital signal processing using approximate adders," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 32, no. 1, pp. 124–137, 2013.

[45] M. Imani, S. Gupta, Y. Kim, M. Zhou, and T. Rosing, "Digitalpim: Digital-based processing in-memory for big data acceleration," in *Great Lakes Symposium on VLSI*, 2019.

[46] M. Imani, S. Gupta, and T. Rosing, "Genpim: Generalized processing in-memory to accelerate data-intensive applications," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2018.

[47] V. Seshadri, Y. Kim, C. Fallin, D. Lee, R. Ausavarungnirun, G. Pekhimenko, Y. Luo, O. Mutlu, P. B. Gibbons, M. A. Kozuch, and T. C. Mowry, "Rowclone: Fast and energy-efficient in-dram bulk data copy and initialization," in *IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2013.

[48] A. Shilov, "Jedec publishes hbm2 specification as samsung begins mass production of chips," Jan 2016. [Online]. Available: https://www.anandtech.com/show/9969/jedec-publishes-hbm2-specification

[49] H. Saadat, H. Bokhari, and S. Parameswaran, "Minimally biased multipliers for approximate integer and floating-point multiplication," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, pp. 2623–2635, 2018.

[50] Z. E. Mamaghani, S. Ullah, and A. Kumar, "Simdive: Approximate simd soft multiplier-divider for fpgas with tunable accuracy," in *30th 2020 ACM Great Lakes Symposium on VLSI (GLSVLSI)*, September 2020.

[51] C. Eckert, X. Wang, J. Wang, A. Subramaniyan, R. Iyer, D. Sylvester, D. Blaaauw, and R. Das, "Neural cache: Bit-serial in-cache acceleration of deep neural networks," in *ACM/IEEE International Symposium on Computer Architecture (ISCA)*, 2018.

[52] C. Huang and I. G. Thakkar, "Mitigating the latency-area tradeoffs for DRAM design with coarse-grained monolithic 3d (M3D) integration," *CoRR*, vol. abs/2008.11367, 2020. [Online]. Available: https://arxiv.org/abs/2008.11367

BIOGRAPHIES

Sathwika Bavikadi received her Bachelor of Technology in Electrical and Communication Engineering from Jawaharlal Nehru Technology University, Hyderabad, India. She then pursued her Master of Science in Electrical Engineering with emphasis on Signal Processing from Blekinge Institute of Technology (BTH), Karlskrona, Sweden. She is currently working on her Ph.D. in Electrical and Computer Engineering at George Mason University (GMU), Fairfax, Virginia, USA. Her research interests include Hardware Acceleration for Machine Learning, Computer Architecture, and In-memory Computing. She is an active student IEEE member.

Purab Ranjan Sutradhar received his Bachelor of Science in Electrical and Electronic Engineering from Bangladesh University of Engineering and Technology, Dhaka, Bangladesh in 2017. He is currently pursuing his Ph.D. in Computer Engineering at Rochester Institute of Technology, Rochester, New York, USA. His research interests include Data-centric and Memory-centric Computing, Deep Learning, and AI Applications. He is an active student IEEE member.

Mark A. Indovina (Senior IEEE Member, 2000) received an A.S in Applied Science, and B.S., and M.S. degrees in Electrical Engineering from the Rochester Institute of Technology, Rochester, NY, USA, in 1979, 1982, and 1987, respectively, where he is currently a Senior Lecturer with the Department of Electrical and Microelectronic Engineering. His research interests span low power analog, mixed-signal, and digital system design for a variety of novel applications. He is also one of the founders of Tenrehte Technologies, Inc., a company that designs and develops energy efficiency products.

Amlan Ganguly is currently an Associate Professor and Department Head in the Department of Computer Engineering at Rochester Institute of Technology, Rochester, NY, USA. He received his Ph.D. and MS degrees from Washington State University, USA, and BTech from Indian Institute of Technology, Kharagpur, India in 2010, 2008, and 2005 respectively. His research interests are in robust and scalable intra-chip and inter-chip interconnection architectures and novel datacenter networks with wireless interconnect as well as non-von Neumann Architectures. He is a senior member of IEEE.

Sai Manoj P D (S'13-M'15) is an assistant professor at George Mason University (GMU). Prior joining to GMU as an assistant professor, he served as a research assistant professor and post-doctoral research fellow at GMU. He received his Ph.D. in Electrical and Electronics Engineering from Nanyang Technological University, Singapore in 2015. His research interests include on-chip hardware security, neuromorphic computing, adversarial machine learning, self-aware SoC design, image processing, and time-series analysis.