Reconfigurable Processing-in-Memory Architecture for Data Intensive Applications

Sathwika Bavikadi*, Purab Ranjan Sutradhar†, Amlan Ganguly†, Sai Manoj Pudukotai Dinakarrao*
*Department of Electrical and Computer Engineering, George Mason University, Fairfax, VA, USA
†Department of Computer Engineering, Rochester Institute of Technology, Rochester, USA.
*{sbavikad, spudukot}@gmu.edu; †{ps9525,axgeec}@rit.edu

***Abstract*—Emerging applications reliant on deep neural networks (DNNs) and convolutional neural networks (CNNs) demand substantial data for computation and analysis. Deploying DNNs and CNNs often leads to resource constraints, data movement overheads between memory and compute units. Architectural paradigms like Processing-in-Memory (PIM) have emerged to mitigate these challenges. However, existing PIM architectures necessitate trade-offs involving power, performance, area, energy efficiency, and programmability. Our proposed solution focuses on achieving higher energy efficiency while preserving programmability and flexibility. We introduce a novel multi-core reconfigurable architecture with fine-grained integration within DRAM sub-arrays, resulting in superior performance and energy-efficiency compared to conventional PIM architectures. Each core in our design comprises multiple processing elements (PEs), standalone processors equipped with programmable functional units constructed using high-speed reconfigurable multi-functional look-up-tables (M-LUTs). These M-LUTs enable multiple functional outputs, such as convolution, pooling, and activation functions, in a time-multiplexed manner, eliminating the need for different LUTs for each function. Special function LUTs provide simultaneous outputs, enabling ultra-low latency parallel processing for tasks like multiplication and accumulation, along with functions like activation, pooling, and batch-normalization required for CNN acceleration. This comprehensive approach enhances efficiency and performance, rendering our reconfigurable architecture suitable for demanding Big Data and AI acceleration applications.**

I. Introduction

Rapid advancements in computer architecture, hardware fabrication, and integration, along with software applications, led to the development of various fields, including computer vision, image processing, artificial intelligence (AI), and natural language processing (NLP). These innovations have led to an eventual increase in demand for enhanced performance, efficiency, and data processing capabilities.

To tackle the inefficiencies of traditional computing methods, machine learning (ML) and deep learning (DL) emerged as powerful solutions for handling large datasets [1], [15]. Innovative architectural paradigms, such as 'non-von Neumann' designs like processing-in-memory (PIM) or In-Memory Computing (IMC), are introduced to alleviate data transfer bottlenecks [12], [15]. IMC architectures perform computations inside memory chips, delivering superior energy efficiency due to intra-memory communication and computations.

Numerous PIM designs are implemented on a wide range of emerging memory technologies. such as traditional volatile static random access memory (SRAM) [10], dynamic random access memory (DRAM) [7]–[9], [14], [19], as well as non-volatile memory technologies like Resistive RAM (ReRAM) [17], and Magnetic RAM (STT/SOT-MRAM) [2]. A recent approach to PIM design leverages memory look-up-tables (LUT) for arithmetic and logical operations within memory chips [9], [18], [19].

However, these existing architectures are specialized for specific applications and operations, lacking adaptability to diverse data-intensive tasks with minimal or no programmability [3]. Furthermore, current LUT-based designs are limited to supporting only one type of functionality, necessitating different LUTs for different functions. While offering performance benefits, this architectural approach suffers from inefficiency in terms of area and latency. Current PIM architectures excel in compute- or memory-intensive applications but fall short when handling other types of tasks [6]. Therefore, there's a need for a versatile hardware platform capable of supporting various CNN/DNN operations to achieve superior energy efficiency, area efficiency, and performance.

To address these challenges and offer a larger degree of functional flexibility and programmability, *we propose a heterogeneously programmed multi-functional look-up-table-based (M-LUT) reconfigurable PIM architecture that supports existing and emerging applications with low overheads and high programmability. We define the LUT multi-functionality as the capability of the LUT to provide output for multiple functions without the requirement of reprogramming.* Furthermore, multiple LUTs can be combined to formulate a different function, i.e., two LUTs that perform shift and add operations (say) can be connected by programming the interconnects to realize a different function, i.e., multiply operation (say), instead of designing a separate LUT for add, shift, and multiply operations. Such an architectural approach not only provides a reduced number of LUTs but also increases the utilization efficiency and functional support offered by LUTs.

From the system perspective, the proposed architecture consists of multiple clusters, as shown in Figure 1. Each cluster comprises Processing Elements (PEs) that encompass three types of M-LUT cores: ALU LUT core, special ALU (S-ALU) LUT core, and special function (SF) LUT core. Each of these cores facilitates multiple functional programmable operations on a pair of 4-bit or a single 8-bit input data. An array of these PEs forms a cluster that can be utilized to implement different layers of CNNs and DNNs for various CNN inference applications. Additionally, three different cluster design exploration is adapted to provide inherent computing support for MAC operations, activation, pooling, and normalization operations.

To summarize, the cardinal contributions of this work are:

- We introduce a novel and flexible in-memory computing architecture by introducing reconfigurable and reprogrammable M-LUTs capable of performing multi-functional operations required to process different neural network layers for CNN acceleration.
- We propose three different kinds of M-LUT cores with different functionality: ALU LUT core, S-ALU LUT core, and SF-ALU LUT core, which are specially designed to tackle multi-functional operations required for diverse CNN acceleration tasks.
- We design three distinct cluster architectures with heterogenous core alignment, supporting a wide range of operations performed across different layers of the CNN for multiple ML applications.

II. Background and Related Works

Processing in Memory, also known as in-memory computing devices, are memory-centric architectures [8], [14] entirely implemented on a memory chip. There exist multiple PIM designs implemented on a wide range of emerging memory technologies, such as traditional volatile static random access memory (SRAM) [10], dynamic random access memory (DRAM) [4], [5], [7]–[9], [14], [19]–[21]. And non-volatile memory such as ReRAM [17], Phase-Change Memory (PCM), Spin-Transfer Torque (STT)-MRAM [2], and Spin-Orbit Torque (SOT)-MRAM [23] technologies.

Numerous IMC hardware accelerators that support ML applications are introduced in the literature [6]. A majority of these IMC works [7]–[9], [14] focus on performing faster computations and do not consider the reconfigurability and networking concerns of the accelerators. However, the functionality of these architectures is almost exclusively limited by their application, reconfigurability, overheads, latency, and inference of CNN/DNNs.

Although there have been several PIM architectures that leverage LUTs for supporting in-memory computations for AI acceleration [9], [11], [13], to the best of our knowledge, these architectures do not offer in-situ reconfigurability through LUT programming. Further, these works leverage relatively larger LUTs, occupying a large computing footprint and thereby diminishing operational parallelism. This motivates us to develop a novel LUT-based in-situ reconfigurable PIM architecture that can support multi-functionality within a relatively smaller computing footprint via LUT reprogramming for accelerating diverse ML algorithms.

III. Proposed Multi-functional LUT-based Reconfigurable Architecture

The hierarchical view of the proposed reconfigurable PIM architecture equipped with M-LUTs is illustrated in Figure 1. The proposed architecture includes the arrangement of clusters inside a DRAM bank, the architecture of a single cluster, space exploration of cores and the router interconnect inside each PE, and the M-LUT core architecture.

This architecture is composed of multiple clusters. Each cluster encompasses Processing Elements (PEs) that contain reconfigurable M-LUT cores, facilitating multi-functional programmable operations on a pair of 4-bit or a single 8-bit input data. We chose this precision as most computer vision applications perform reliably at this precision with minimal accuracy loss compared to higher precision [22].

A. Cluster Architecture

The primary goal of the proposed architecture is to implement complex operations while incurring the least amount of complexity to the hardware design. This is carried out by executing multi-stage operations inside the PE consisting of M-LUT cores coupled together with a router mechanism. Each PE can be programmed to perform a wide range of operations such as multiply and accumulate, substitution, comparison, bit-wise logic operations, hyperbolics, sigmoid, and ReLU activation and pooling operations. Therefore, we propose an architecture supporting an array of these PEs to form a cluster that can be utilized to implement different layers of CNNs such as convolutional, fully-connected, activation, and pooling.

To support various CNNs and DNN layers, our architecture allows for the integration of diverse PEs within a cluster. To address the needs of convolutional and fully-connected layers that perform convolution operations that fundamentally involve matrix multiplications, implemented as a chain of Multiplication and Accumulation (MAC) operations, we introduce Multiply and Accumulate Processing Element (MAC PE). On the other hand, to support a wide variety of operations performed in the Activation, Pooling, and Normalization layers, we introduce the Special Function Processing Element (SF PE). Therefore, the proposed architecture supports an array of these PEs (MAC PE, SF PE) to form a cluster that can be utilized to implement different layers of CNNs and DNNs. These PEs are interconnected by the router, forming a 2-D array inside the memory bank, as shown in Figure 1. The close proximity to the memory bank allows quick memory access and the ability to perform various in-memory operations. These operations are carried out by executing multi-stage operations on the PEs inside a cluster, with the help of the router.

Different cluster designs are proposed for the proposed architectures to support diverse operations performed in different layers of the CNN. Its functionality is discussed below:

Cluster Architecture 1: This cluster supports the design exploration of the 9 PEs in a 3x3 manner, demonstrated in Figure 1. This cluster architecture comprises 8 MAC PEs and one SF PE and can support 8 MAC operations and a special function operation at the same time. This arrangement of clusters is adapted to support smaller-scale MAC operations, which can be implemented mainly in Fully Connected layers.

Cluster Architecture 2: In order to scale up the size of the operands, we aggregate 25 PEs, arranged in a 5×5 formation, to form a cluster. Therefore, this cluster supports the design exploration of the 25 PEs in a 5x5 grid manner. This cluster architecture comprises 24 MAC PEs and one SF PE and can support 24 MAC operations and a special function operation at the same time. This arrangement of clusters is adapted to support wider, smaller-scale MAC operations, which can be implemented in the latter convolutional operation layers and fully connected layers.

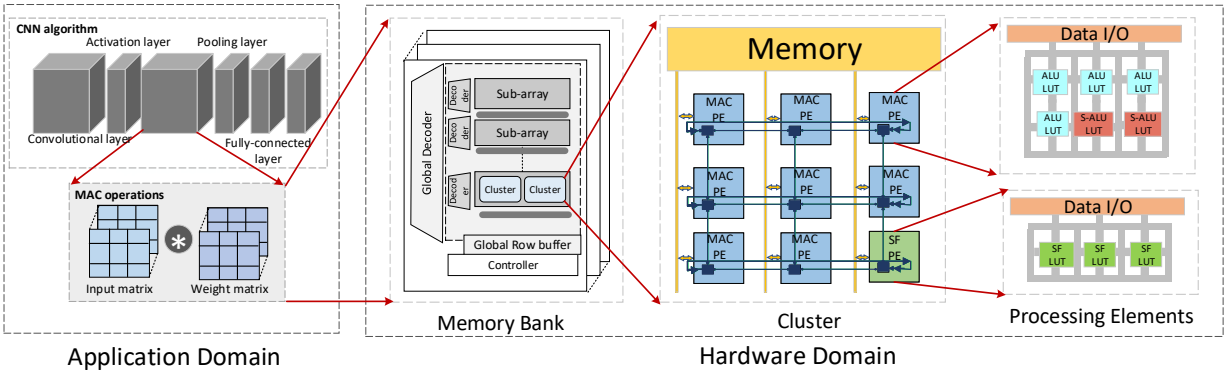

Fig. 1. Hierarchical Architecture showing the cluster arrangement and M-LUT core organization inside the processing elements

Cluster Architecture 3: To further scale up the size of the operands, we combine 49 PEs into a cluster placed in a 7×7 arrangement. As a result, this cluster facilitates the 7x7 grid-based design exploration of the 49 PEs. This cluster architecture comprises 48 MAC PEs and one SF PE and can support 48 MAC operations and a special function operation at the same time. This arrangement of clusters is adapted to support wider large-scale MAC operations, which can be implemented in the compute-intensive convolutional layers.

B. Processing Element Architecture

The Processing Element in the proposed architecture is an independent processing unit capable of performing complex operations with 8-bit fixed point precision by organizing a series of micro-operations across the M-LUT cores in several operational stages (discussed in detail in Section IV). The M-LUT core design explorations in a PE aim to facilitate intrinsic computational support to perform MAC operations, activation, and pooling operations for ML acceleration within different neural network layers.

The proposed architecture comprises two different PEs, a MAC PE and SF PE, comprising different sets of reconfigurable M-LUTs cores (ALU-LUT, S-ALU-LUT, SF-LUT). Unlike the LUT cores in existing works, the proposed M-LUT cores are capable of performing distinct operations from each other and can provide multiple outputs corresponding to multiple functionalities in a multiplexed manner, thereby called multi-functional LUTs. This approach not only provides a reduced number of LUTs but also increases the utilization efficiency and functional support offered by LUTs.

C. Core Programming Inside the Processing Elements

This section discusses the functionalities of the cores, referring to corresponding color codes in Figure 1.

*1) **MAC Processing Element (MAC PE):*** We propose a single PE called Multiply and Accumulate Processing Element to mainly support multiplication and accumulation operations by utilizing the multi-functionality of M-LUT cores. The MAC PE comprises two different reconfigurable M-LUT cores (ALU-LUT, S-ALU-LUT) that are grouped together and interconnected by a router. The MAC PE consists of a total of six M-LUT cores (4 ALU-LUT cores, 2 SALU-LUT cores).

ALU-LUT core (Core 1 to 4): The ALU-LUT cores are specifically programmed to implement the MAC operations in the PIM. The blue squares in Figure 1 represent the multi-functional LUT cores that are programmed to perform 4-bit AND or XOR operations on a pair of 4-bit data inputs and generate a 4-bit output. A multiplexer is used to select the functionality required for the different operations of the CNN algorithm to perform either XOR or AND operation on the inputs, as shown in Figure 2 (a).

S-ALU LUT core (Core 5 and 6): The second kind of M-LUT core used in the MAC PE, represented in red squares in Figure 1, called the special-ALU LUT core. The MAC PE contains two of these cores that are programmed such that the output consists of two entirely different operations (XOR and AND) on the same pair of inputs. Despite the fact that the S-ALU-LUT core supports the same operations (XOR and AND) as the ALU-LUT core, its functionality is entirely different. This core is used in a special scenario when we need both XOR and AND operations for the same input data, mainly used for the accumulation process. This core is programmed to produce 8-bit output data for a pair of 4-bit inputs. The upper half of the core output represents the 4-bit XOR operation of the input data while the lower half represents the 4-bit AND operation of the same input data as shown in Figure 2 (b). Thus, without the need to create separate LUT cores for various purposes, this unique M-LUT called S-ALU LUT core may deliver several outputs pertaining to different functionalities concurrently.

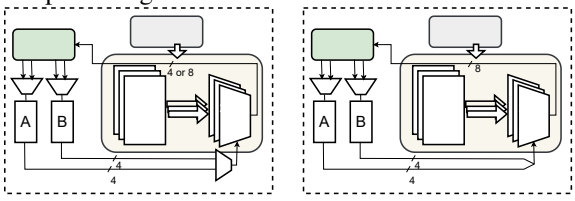

Fig. 2. Microarchitectures of Reconfigurable M-LUT-based PIM cores (a) ALU-LUT core and SF-LUT core (b) S-ALU-LUT core

*2) **SF Processing Element (SF PE):*** We propose a single PE called a special function PE that is capable of performing a wide range of special function operations such as activation, pooling, and batch normalization operations by utilizing the M-LUT cores. Due to the considerably lesser complexity of these operations, they are all carried out in the same SF PE. The cluster contains only one of these SF PE, consisting of 3 special function LUT cores called SF-LUT cores.

SF-LUT core (Core 1 to 3): The third kind of M-LUT core used in the proposed architecture is a special function LUT core, represented in the green square in Figure 1. SF-LUT is programmed to perform 8-bit special-function operations such as pooling, batch normalization, and activation operations, including different activations such as sigmoid, hyperbolic, and

ReLU operations.In Figure 2 (a) SF LUT core is implemented using 8-bit 256-to-1 multiplexers. For example, in order to perform an activation operation with an 8-bit operand, the 8-bit MUX in the PIM core is used to perform a look-up operation and provide 8-bit output.

IV. Operations Supported by the Proposed Reconfigurable Architecture

Majority of the operations performed for ML acceleration are essentially translated into a large set of vector or matrix multiplications (known as GEMV or GEMM). Furthermore, matrix multiplication can be performed in the form of repeated MAC operations in the MAC PEs. Similarly, the Convolutional operations and Fully Connected operations can be executed in the MAC PE. Whereas the SF PEs are designed to implement special function operations such as non-linear activation, pooling, and normalization operations using the memory look-up approach. These operations are carried out within the cluster by executing a multi-stage pipeline of the M-LUT cores inside the PEs coupled together with a routing mechanism.

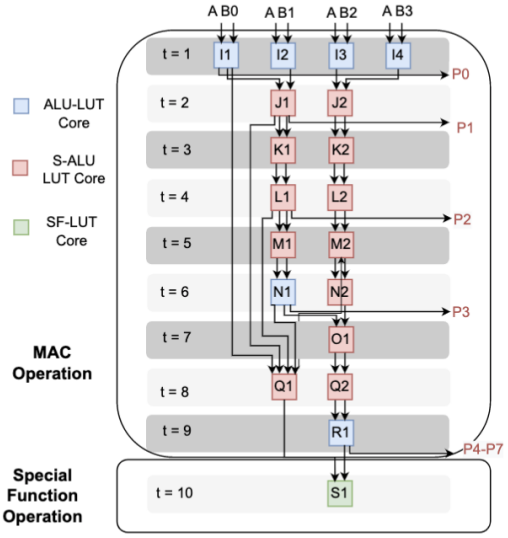

Fig. 3. Overview of the dataflow for MAC operation in the proposed Reconfigurable PIM architecture

A. MAC Operations Supported by the proposed architecture

Each M-LUT core is capable of performing operations on a pair of 4-bit, or a single 8-bit operand. For MAC operations, we process a pair of 4-bit data in parallel to produce 8-bit output using the ALU-LUT and S-ALU LUT cores.

To execute the MAC operation on two 4-bit data operands, initially, both the input data, A and B are split into sections A_3, A_2, A_1, A_0, and B_3, B_2, B_1, B_0 respectively. The 4-bit multiplication is performed similarly to decimal multiplication. As demonstrated in Figure 3, a routing mechanism is used to perform the MAC operation in a multi-stage pipeline.

Figure 3 also illustrates how each process in the dataflow has been assigned a special tag consisting of a letter and a number for ease of implementation and testing. Numbers 0, 1, 2, and 3 denote various parallel operations carried by the cores in each clock cycle, whereas letters I, J, K, L, M, N, O, Q, R, and S denote the clock steps of M-LUT operations, P0-P7 represent the MAC operation output of the MAC PE. During the runtime, P0-P7 of the MAC operation is accumulated using the S-ALU LUT core. This accumulated output is later passed to the SF-LUT core in SF PE to perform either activation, pooling, or normalization operations, which can be carried out in a single clock cycle.

B. Special Function Operations Supported by the proposed architecture

The output of the MAC operation from the MAC PE is passed to the multi-functional SF LUT core in SF PE to implement either of the special function operations such as activation, pooling, and normalization operations. A multiplexer is used to select the different special function operations to be implemented in the SF LUT. Based on the input from the multiplexer, the multi-functional M-LUT core performs either sigmoid, hyperbolic, or ReLU activation operations or max pooling, average pooling operations, or normalization operations on the input data. This operation can be performed in a single clock cycle during the execution. The router is used to enable the chain of operations required for MAC and activation operations inside the cluster.

The key advantage of the proposed architecture is that it enables a routing scheme and parallelization process to efficiently utilize the cores inside the cluster. Moreover, it can be said that the M-LUTs in the proposed architecture are capable of reprogramming at run-time to perform complex computational operations to implement CNN at ultra-low latency. The cluster is capable of executing complicated arithmetic or logical tasks over a single or multiple stages by combining the capability of multiple PE functionalities.

V. Evaluation

A. Design Verification

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 using 28 nm standard cell library from TSMC and are presented in Table I.

TABLE I
Characteristics of Reconfigurable Multi-Functional Hardware Accelerator and its Components in 28 nm Technology Node

Component	Delay (ns)	Power (mW)	Active Area(μm^2)
ALU-LUT Core	0.10	0.00177	8010
S-ALU-LUT Core	0.26	0.00497	13210
SF-LUT Core	0.7	0.01853	141304
MAC PE	0.92	0.01702	58460
SF PE	0.7	0.05559	1461500
Proposed PIM Cluster 1	1.62	0.19175	526140
Proposed PIM Cluster 2	1.62	0.46407	1461500
Proposed PIM Cluster 3	1.62	0.87255	2864540
LUT Core [19]	0.8	2.7	4196.64
LUT Cluster (MAC Operation) [19]	6.4	8.2-11	37769.81
Intra-Subarray Communication [16]*	63.0	0.028 μJ/comm	N/A
Inter-Subarray Communication [7] 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

Firstly, it is observed that due to the different operational support provided by MLUT cores, they have different delay, area, and power metrics. SF-LUTs, which process 8-bit data on 8-bit memory LUTs, have the highest delay, area, and power consumption. In contrast, ALU LUT cores, designed

for processing a pair of 4-bit data on 4-bit memory LUTs, exhibit the least delay, area, and power consumption. Due to the same reason, MAC PE shows lower delay, area, and power consumption than SF PE. However, compared to the LUT core [19], the proposed cores have lower delay and power consumption, but the active area is 2× greater.

Table I also presents the cluster characteristics for different design explorations discussed in Section III-A. As the cluster size (Cluster 1, 2, 3) increases in terms of the number of PEs (9, 25, 49), both area and power consumption increase. Clusters 1, 2, and 3 can simultaneously perform 8-bit MAC and activation operations, resulting in similar delay values. Specifically, when performing an 8-bit MAC operation and activation operation on the proposed architecture, the delay is 1.62 ns, whereas the LUT core [19]] requires 6.4 ns for MAC operation alone, making the proposed architecture nearly 4 × faster. Therefore, it is observed that the proposed architecture is highly suitable for ultra-low latency, low-power applications such as real-time IoT devices and edge devices.

In order to perform an 8-bit MAC operation on a prior LUT-based PIM architecture such as LACC [9], it requires 1048576 (2^{16} × 16) pre-computed results of an operation. In comparison, the proposed architecture requires 18432 (6 × 2^8 × 8) pre-computed results of an operation by using the six 4-bit M-LUT cores in the MAC PE. This leads to 86× area efficiency than LACC [9]. Despite requiring 9 clock cycles compared to a single clock cycle in LACC PIM, the proposed architecture's parallel processing capability allows it to perform multiple tasks simultaneously, contributing to lower dynamic power consumption.

B. Performance Evaluation

In this subsection, we perform a comparative performance analysis of the proposed architecture in terms of throughput (Frames per second) and energy efficiency (Frames per Joule) on LeNet, AlexNet, ResNet-18, -34, and -50.

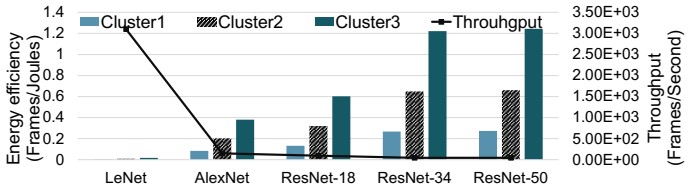

Fig. 4. Comparison of Energy efficiency (Frames/Joules) and Throughput (Frames/second) for LeNet, AlexNet, ResNet18, ResNet34, and ResNet50 on the proposed architecture.

Figure 4 illustrates the relationship between energy efficiency and the depth of CNN algorithms. As the number of layers in the CNN increases, more MAC, activation operations are required, leading to increased parallelization. Consequently, CNNs with a higher number of layers achieve higher energy efficiency. It is observed that LeNet, AlexNet, and ResNet 18 achieved the inference energy efficiency of 0.0011 Frames/Joule, 0.024 Frames/Joule, and 0.038 Frames/Joule respectively.

The proposed architecture achieves better performance for CNN algorithms with comparatively lower computational workloads such as LeNet, shown in Figure 4. However, for 8-layered AlexNet, the proposed architecture achieves an inference throughput of 150.3 Frames/s, while the 50-layered ResNet algorithm attains an inference throughput of 45.9 Frames/s. Therefore, it can be said that the proposed architecture can achieve impressive performance while implementing complex CNN/DNNs, operations due to its robust parallel processing ability. For instance, ResNet-50, the largest network implemented on the proposed architecture consists of 50 layers with thirty-eight billion computations that can be processed within 10 ms on the proposed architecture.

Due to the similar design exploration and distribution of the tasks in the clusters, a similar trend in terms of energy efficiency is observed for all three cluster implementations for all the CNNs. However, due to the simultaneous parallel operational support of all the PEs in the cluster, the throughput remains the same for all three cluster implementations.

C. Inference Accuracy

We evaluate our proposed architecture (Cluster 1) for various state-of-the-art deep neural networks such as LeNet, AlexNet [1], ResNet -18,-34,-50, on MNIST and CIFAR-10 datasets. Figure 5 shows the Top-5 accuracy comparison plots for 16-bit floating-point (FP), and 8-bit fixed-point data precision (both inputs and weights).

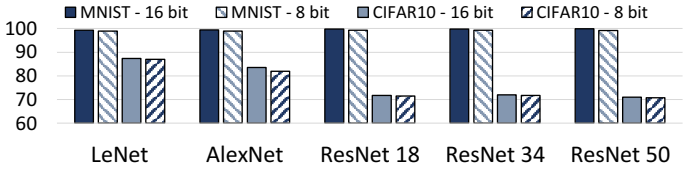

Fig. 5. Comparison of Top-5 accuracies of LeNet, AlexNet, ResNet-18, -34 and -50 on MNIST, CIFAR-10 dataset for 16-bit, 8-bit data precision

It is observed that the accuracies obtained on the evaluated networks are very similar for 16-bit and 8-bit precision data. The Top-1 accuracy obtained for the MNIST dataset when implemented on AlexNet is 98.89% and 99.43% for 16-bit and 8-bit precision, respectively. On the other hand, the Top-1 accuracy obtained for the CIFAR-10 dataset when implemented on AlexNet is 83.5% and 82% for 16-bit and 8-bit precision, respectively. It is also observed that the CNN accuracies for the CIFAR-10 dataset are noticeably lower when compared to the MNIST dataset, with a performance degradation of around 10%-15% for all the CNNs deployed. This difference in accuracy can be attributed to the higher complexity of the CIFAR-10. It's important to mention that while higher accuracy with CIFAR-10 has been reported in the literature, those results typically employ higher data precision than what we adopted in this paper [22].

D. Performance Comparison with State-of-the-Art Hardware Accelerators for CNN Implementation

Performance is evaluated by comparing the proposed architecture with state-of-the-art PIM accelerator architectures in terms of power consumption (Watt) and throughput (Frames/second), as shown in Figure 6.

As a proof of concept, we evaluate and implement AlexNet [1] on the proposed architecture with 8-bit precision.We compared our architecture with several PIM architectures, including DRAM-based bulk bit-wise processing devices DRISA

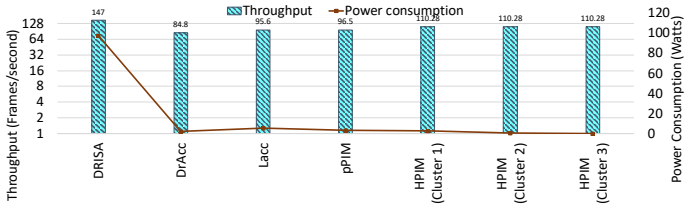

Fig. 6. Comparative performance analysis of proposed architecture with respect to previous PIM architectures in terms of throughput (Frames/second) and power consumption (Watt).

[14], DrAcc [8], LUT-based PIM implemented on the DRAM platforms such as LAcc [9], and pPIM architecture [19].

Among the PIMs studied here, a relatively higher throughput is observed for DRISA [14] due to its ability to parallelize operations across multiple banks. DrAcc [8] achieves higher performance with 8-bit ternary precision inferences, similar to pPIM [19]. LAcc [9], pPIM [19] demonstrate the advantages of using LUTs for utilizing pre-calculated results, achieving impressive inference performances.

It can be observed that the throughput stays constant for all three cluster implementations due to the simultaneous parallel operational support provided by all of the PEs in the cluster. However, our architecture, utilizing reconfigurable multi-functional M-LUTs for CNN algorithms, achieves notably higher AlexNet throughput compared to the LUT-based PIMs in the comparison. Furthermore, it boasts significantly higher throughput than other PIM architectures like DRISA, Dracc, and Neural cache, as shown in Figure 6. A similar trend is observed in for power consumption comparison, the proposed architecture demonstrates lower power consumption compared to the PIM architectures. It also outperforms LAcc and pPIM by approximately 1.14 $\times$ for AlexNet inference throughput. Additionally, for Cluster architectures 1, 2, and 3, our architecture achieves 2.4 $\times$, 10 $\times$, and 101 $\times$ higher energy efficiency than LAcc and pPIM implementations, respectively, for AlexNet inference.

VI. Conclusion

In order to address the energy efficiency and flexibility requirements for computer architectures, we present a novel multi-functional LUT-based reconfigurable PIM architecture in this work. The architecture is tailored for CNN/DNN inference applications, supporting existing and emerging applications with minimal overheads and high programmability. In performance evaluations compared to state-of-the-art PIM architectures, our design significantly outperforms a DRAM-based LUT-based PIM architecture., with 101 $\times$ higher energy efficiency and 1.14 $\times$ higher throughput when implementing AlexNet. Although the proposed architecture is primarily designed for CNN acceleration, its multi-functionality, reconfiguration, and ultra-low latency implementation make it suitable for a wider range of application domains, such as real-time IoT, edge devices, mobile applications, and automated systems.

VII. Acknowledgements

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.

References

[1] M. Z. Alom *et al.*, "The history began from alexnet: A comprehensive survey on deep learning approaches," *arXiv*, 2018.

[2] S. Angizi *et al.*, "Mrima: An mram-based in-memory accelerator," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 5, pp. 1123–1136, 2020.

[3] S. Bavikadi *et al.*, "A survey on machine learning accelerators and evolutionary hardware platforms," *IEEE Design & Test*, vol. 39, no. 3, pp. 91–116, 2022.

[4] S. Bavikadi *et al.*, "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.

[5] S. Bavikadi *et al.*, "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.

[6] S. Bavikadi *et al.*, "A review of in-memory computing architectures for machine learning applications," ser. GLSVLSI '20, 2020.

[7] K. K. Chang *et al.*, "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.

[8] Q. Deng *et al.*, "Dracc: a dram based accelerator for accurate cnn inference," in *2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC)*, 2018, pp. 1–6.

[9] Q. Deng *et al.*, "Lacc: Exploiting lookup table-based fast and accurate vector multiplication in dram-based cnn accelerator," *2019 56th ACM/IEEE Design Automation Conference (DAC)*, pp. 1–6, 2019.

[10] C. Eckert *et al.*, "Neural cache: Bit-serial in-cache acceleration of deep neural networks," in *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*, 2018, pp. 383–396.

[11] J. D. Ferreira *et al.*, "pluto: In-dram lookup tables to enable massively parallel general-purpose computation," *CoRR*, vol. abs/2104.07699, 2021.

[12] A. Ganguly, R. Muralidhar, and V. Singh, "Towards energy efficient non-von neumann architectures for deep learning," in *Int. Symp. on Quality Electronic Design (ISQED)*, 2019.

[13] P. Gu *et al.*, "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*, pp. 1–1, 2020.

[14] S. Li *et al.*, "Drisa: A dram-based reconfigurable in-situ accelerator," in *2017 50th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2017, pp. 288–301.

[15] S.-L. Lu *et al.*, "Scaling the "memory wall": Designer track," in *2012 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2012, pp. 271–272.

[16] V. Seshadri *et al.*, "Rowclone: Fast and energy-efficient in-dram bulk data copy and initialization," in *46th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, Dec 2013, pp. 185–197.

[17] L. Song *et al.*, "Pipelayer: A pipelined reram-based accelerator for deep learning," in *2017 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, 2017, pp. 541–552.

[18] P. R. Sutradhar *et al.*, "Look-up-table based processing-in-memoryarchitecture with programmable precision-scalingfor deep learning applications," *IEEE TPDS*, 2021.

[19] P. R. Sutradhar *et al.*, "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.

[20] P. R. Sutradhar *et al.*, "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.

[21] P. R. Sutradhar *et al.*, "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.

[22] K. Vasquez *et al.*, "Activation Density based Mixed-Precision Quantization for Energy Efficient Neural Networks," *arXiv e-prints*, p. arXiv:2101.04354, Jan. 2021.

[23] G. Yuan *et al.*, "A sot-mram-based processing-in-memory engine for highly compressed dnn implementation," *ArXiv*, vol. abs/1912.05416, 2019.