pPIM: A Programmable Processor-in-Memory Architecture With Precision-Scaling for Deep Learning

Purab Ranjan Sutradhar, *Student Member, IEEE*, Mark Connolly, Sathwika Bavikadi, Sai Manoj P D, *Member, IEEE*, Mark A. Indovina, *Senior Member, IEEE*, and Amlan Ganguly, *Senior Member, IEEE*

Abstract—Memory access latencies and low data transfer bandwidth limit the processing speed of many data intensive applications such as Convolutional Neural Networks (CNNs) in conventional Von Neumann architectures. Processing in Memory (PIM) is envisioned as a potential hardware solution for such applications as the data access bottlenecks can be avoided in PIM by performing computations within the memory die. However, PIM realizations with logic-based complex processing units within the memory present complicated fabrication challenges. In this letter, we propose to leverage the existing memory infrastructure to implement a programmable PIM (pPIM), a novel Look-Up-Table (LUT)-based PIM where all the processing units are implemented solely with LUTs, as opposed to prior LUT-based PIM implementations that combine LUT with logic circuitry for computations. This enables pPIM to perform ultra-low power & low-latency operations with minimal fabrication complications. Moreover, the complete LUT-based design offers simple 'memory write' based programmability in pPIM. Enabling precision scaling further improves the performance and the power consumption for CNN applications. The programmability feature potentially makes it easier for online training implementations. Our preliminary simulations demonstrate that our proposed pPIM can achieve 2000x, 657.5x and 1.46x improvement in inference throughput per unit power consumption compared to state-of-the-art conventional processor architecture, Graphics Processing Unit (GPUs) and a prior hybrid LUT-logic based PIM respectively. Furthermore, precision scaling improves the energy efficiency of the pPIM approximately by 1.35x over its full-precision operation.

Index Terms—Processing in memory, look up table, convolutional neural network, deep neural network, DRAM

1 Introduction

MEMORY-CENTRIC processing is an emerging area of research which is gaining an increasing attention due to its ability to address the memory-processor communication bottleneck, popularly known as the 'Memory Wall' [1]. Highly memory-intensive applications such as Convoluted Neural Networks (CNN) and Deep Neural Networks (DNN) demand massive parallel computations to be performed at a very low latency. Processing-in-memory (PIM) is envisioned as a desirable alternative to eliminate the power and performance bottleneck of the traditional Von Neumann architectures as it enables computation either in or near data. However, due to implementation constraints, only small processing units capable of simple computations are featured in PIM architectures. On the other hand, image processing and Deep Learning (DL) applications such as CNNs require relatively simple arithmetic calculations on massive amounts of data. Therefore, PIMs capable of intense parallelization of simple computational tasks that largely eliminating the power-delay cost of data movement between the processors and the memory elements are suitable for DL applications.

Attributing processing abilities to the memory can be performed in different ways such as logic-based processor implementation [2], bulk bit-wise in-situ processing with specialized logic circuits [3], [4], [5], [6], LUT-based processor implementations [7] and 3-D implmentations [8]. In this paper, we propose a programmable PIM (pPIM) architecture based on Look-UP-Tables (LUTs) within a Dynamic Random-Access Memory (DRAM) chip. LUT-based PIMs [7] have been shown to be significantly faster and energy-efficient compared to bit-wise logic circuit based architectures [3], [4], [5] due to the absence of switching power consumption of logic gates. Moreover, LUTs of the pPIM provide functional flexibility to implement different types of computations necessary in DL applications such as linear algebraic operations, different activation functions and pooling. Data movements among the pPIM processing nodes are facilitated by existing data movement mechanisms in DRAM with minimal modifications for faster operation and ease of adoption.

The LUT based approach also enables us to support approximate computing through precision-scaling. The pPIM is capable of operating on scaled 4-bit approximate representations of the operands, resulting in further lowering of latency and power consumption without significant compromise of the accuracy of the CNN applications. The baseline pPIM is capable of achieving a frame rate of 96.5 fps at a power consumption of 3.35 W for AlexNet.

2 pPIM Architecture

The PIM architecture proposed in this work is designed to perform data-intensive applications such as CNNs and DNNs. This architecture is presented in a hierarchical view in Fig. 1. At the center of this architecture is the proposed pPIM core, which facilitates programmable operations on two 4-bit inputs. Nine of the pPIM cores are grouped together to form a pPIM cluster that can perform multiple operations such as Multiply-and-Accumulate (MAC) and activation functions on 8-bit operands. These clusters are arranged in rows across DRAM subarrays, forming an overall 2-D array of clusters across a DRAM bank. An array of these pPIM clusters can be used to perform CNN or DNN computations.

2.1 pPIM Core

In order to offer a larger degree of functional flexibility and programmability, an LUT-based design is adopted for the pPIM core instead of a pre-defined logic circuit. Moreover, it has been shown in recent works such as [7] that an LUT-based, in-memory arithmetic unit can perform multiplications with significantly lower delay compared to bitwise computing [3], [4], [5], [6] without any trade-off in accuracy. Our proposed pPIM with LUT-based approach provides the ability to set its functionality to any arbitrary operation. The LUTs are implemented with 8-bit 256-to-1 multiplexers, as shown in Fig. 1. The 8-bit MUX is necessary to accommodate 8-bit operations such as the multiplication of two 4-bit operands. The MUX select lines are controlled by two 4-bit inputs which serve as the operands, as shown in Fig. 1. The inputs to the MUX, called *function-words*, are directly read from a register file located inside the pPIM core. Therefore, switching between functionalities can be achieved simply by reading new a *function-word* from the register file into the MUX inputs. Additional *function-word*s can be accessed from the DRAM subarray when necessary. The inputs to the pPIM cores are received through the interconnection *Router* of the pPIM cluster. These can be *data-word*s stored in the DRAM subarray or from the chip I/O (in case of

Q1
- *Purab Ranjan Sutradhar, Mark Connolly, Mark A. Indovina, and Amlan Ganguly are with the Rochester Institute of Technology Kate Gleason College of Engineering, Rochester, NY. E-mail: {ps9525, mfc5867, maieee, axgeec}@rit.edu.*
- *Sathwika Bavikadi and Sai Manoj P D are with the Electrical and Computer Engineering, George Mason University, Fairfax, VA. E-mail: sbavikad@gmu.edu, saimanoj.p.2013@ieee.org.*

Manuscript received 25 Mar. 2020; accepted 16 July 2020. Date of publication 0 . 0000; date of current version 0 . 0000.
(Corresponding author: Sai Manoj P D.)
Digital Object Identifier no. 10.1109/LCA.2020.3011643

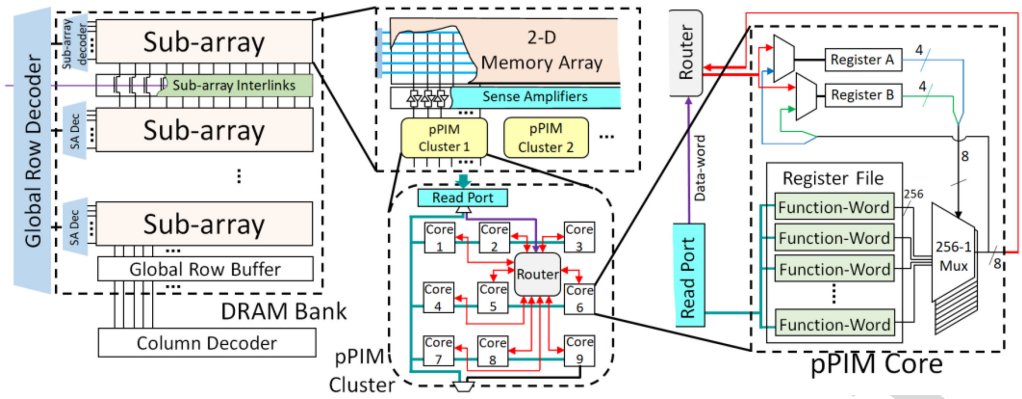

Fig. 1. Hierarchical pPIM Architecture showing pPIM cluster arrangement in a DRAM bank, Core organization inside a cluster and the Core microarchitecture respectively from the left to the right.

direct streaming data), the output from another PIM cluster or another pPIM core. This communication method of the pPIM cores is discussed in detail in the next subsection.

2.2 pPIM Cluster With Precision-Scaling

The pPIM clusters are formed by having nine pPIM cores arranged in 3 x 3 2-D grid, with a view to scaling up the size of the operands. In the case of CNNs or DNNs, a MAC operation is the most expensive and the most frequently used arithmetic function in a convolutional layer. Therefore, here we discuss a use-case where a cluster can be efficiently used to perform MAC operations on 8-bit operands. We choose 8-bit operands in our use-case as it represents the majority of image & video pixel data.

In order to achieve this, the 8-bit MAC instruction is disintegrated into a series of 4-bit operations that can be performed by individual pPIM cores. The 8-bit multiplication is disintegrated into four 4-bit multiplication and nine 4-bit addition operations. We define partial products V_x $(x = 0, 1, 2, 3)$ to be obtained through multiplication of the two 8-bit inputs, 'a' and 'b', where each input is split into its upper and lower four bit segments. Subscripts 'H' and 'L' refer to the upper and lower segments respectively:

$$V_0 = a_L \times b_L \quad (1)$$
$$V_1 = a_L \times b_H \quad (2)$$
$$V_2 = a_H \times b_L \quad (3)$$
$$V_3 = a_H \times b_H, \quad (4)$$

Aggregation of the individual partial products from (1), (2), (3), (4)above is achieved in 8 stages of accumulation by adopting the data flow shown in Fig. 2. Data communications among the pPIM cores within a cluster are achieved through the *Router* which is capable of establishing parallel connections among all cores. It consists of eighteen 8:1 4-bit MUXes which together form a SPIN interconnection fabric. In order to increase the performance of the pPIM, we further propose the use of approximate computing through precision scaling of operands. This is obtained by aggressive truncation of 8-bit operands down to only the 4 most significant bits (MSBs). The MAC operation for precision-scaled 4-bit operands require only 4 stages of operation in total for the product to be accumulated after the multiplication is achieved from (4) using only the most significant 4 bits (a_H & b_H) of the operands. This reduces the MAC operation execution time by half. This precision-scaled approach is evaluated against the baseline 8-bit approach in terms or performance, power and accuracy in Section 3.

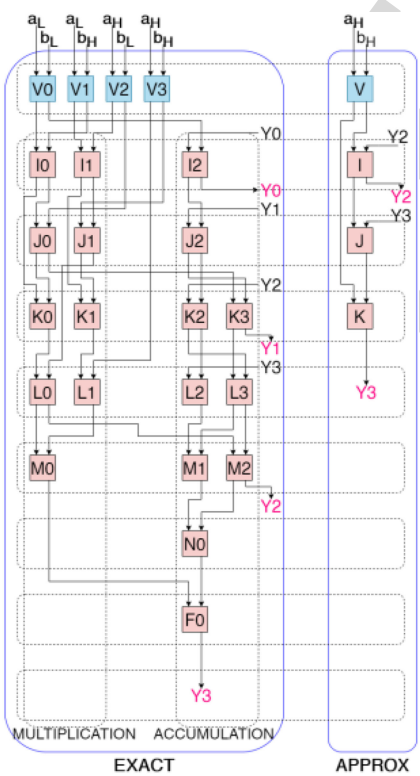

Fig. 2. Sequential model of 8-bit MAC (exact) operation and 4-bit precision-scaled (approx.) MAC. In the figure, blue and red boxes represent cores performing 4-bit multiplication and 4-bit addition respectively. The left and right arrows coming out of each box represent the upper and lower 4-bit results of the core's operation respectively.

2.3 PIM Chip Architecture – Integration With Memory and Data Communication Support

A 2-D array of pPIM clusters is implemented on the memory chip in one memory bank where each cluster can performs at least one independent operation (i.e., MAC). The clusters are arranged along subarrays and are interfaced to the sense-amplifiers whereby they can communicate to the memory bit lines to perform read and write operations. The mapping of the data onto the proposed pPIM is performed as a weight stationary approach i.e., by loading the weights onto the pPIM, one can feed in the input data to obtain the output. Data communication is carefully limited to vertical movements so that the source and the destination of the communications are always located on the same bitlines, either in the same or different subarrays. If the source and the destination clusters are located in the same memory subarray but different rows, then a row copying mechanism called RowClone [9] is utilized in which the entire row of memory is loaded into the subarray row buffer. The row is then written into the destination row(s) from the row buffer.

TABLE 1
Characteristics of PIM Components in 28 nm Technology Node

Component	Delay (ns)	Power(mW)	Active Area (μm^2)
PIM Core	0.8	2.7	4616.85
PIM Cluster (MAC Operation)	6.4	5.2	41551.66
Intra-Subarray Communication (RowClone [9])*	63.0	0.028 μJ/comm	N/A
Inter-Subarray Communication (Lisa–RISC [10]) 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 28 nm technology node.

If source and destination rows are located in different subarrays, then an access-transistor based modification, as proposed in LISA [10], is adopted to facilitate fast data transfer. For complete parallelization of operations such as matrix manipulations required by CNNs and DNNs through data re-use, multicasting of data is essential. Multicasting is facilitated by writing the subarray row buffer to multiple destination rows. This is implemented by a custom control command from the memory controller, inspired from a similar mechanism presented in AMBIT [3], which enables multicasting to specific destination rows using only one additional custom row decoder. By resizing the sense-amplifiers, multicasting can be expanded to more than three rows. Since the adopted mechanism of RowClone [9] for intra-subarray data transfer is faster than LISA [10] for inter-subarray communication, we propose to enlarge the subarrays as much as possible in order to accommodate maximum number of pPIM clusters in the same subarray. To avoid longer bitlines for larger subarrays, we envision increased number of columns rather than increased number of rows for enlarging the subarrays. This can be implemented by having fewer but larger memory banks in the DRAM. These modifications primarily optimize the performance of the PIM architecture [5], rather optimizing the memory organization itself.

Due to programmability of the pPIM cores and hence that of the clusters as well, the proposed pPIM fabric can perform CNN or DNN applications involving matrix multiplications or filtering when disintegrated into a series of parallel MAC operations. Moreover, by appropriate programming of the *function-words*, pPIM clusters can also perform more CNN or DNN operations such as pooling and activation functions.

3 Performance Evaluations

In this section, we evaluate the pPIM in terms of performance, energy consumption and area for DL applications.

3.1 pPIM Core and Cluster Characteristics

The delay, power and area for the pPIM core and cluster are obtained from Synopsys Design Compiler using 28 nm standard cell libraries from TSMC and are presented in Table 1. To reduce the area overhead, the LUT MUXes are implemented using Transmission Gates (TG). The delay of a single 8-bit MAC performed within a cluster involves computations inside the PIM cores as well as communication among the cores. 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 from [9] and [10].

3.2 Performance Evaluation With DL Applications

We present the comparison of the proposed pPIM architecture with a state-of-the-art CPU: Intel Knights Landing (KNL), GPU: Nvidia Tesla P100 along with bulk bit-wise computation based PIM architectures such as the DRAM based DRISA [5], DrAcc [6] & SRAM based Neural Cache [4] and finally, another LUT based PIM, LAcc [7] in terms of power consumption and throughput for CNN inference in Fig. 3a. We find that all the PIMs outperform both CPU and GPU architectures due to the absense of memory access overheads. Neural Cache is the slowest among the PIMs studied here due to its bit-serial nature of computing. pPIM achieves higher performance compared to LAcc due to pPIM's finer (4-bit) granularity of operations that results in smaller sized LUTs. Moreover, in-memory XOR opration based mechanism for adding the partial products adopted in LAcc is more expensive both in power and area compared to additions within the LUTs themselves in pPIM. DRISA has higher throughput than both of the LUT-based PIMs due to its ability to parallelize operations along multiple memory banks. Howeover, this also causes higher power consumption in DRISA compared to both LAcc and pPIM. The higher throughput of DrAcc can be attributed to its use of ternary weighted convolutions that reduce MACs to simpler addition operations. A higher size pPIM-512 with 512 clusters can improve the throughput significantly, albeit at increased power consumption.

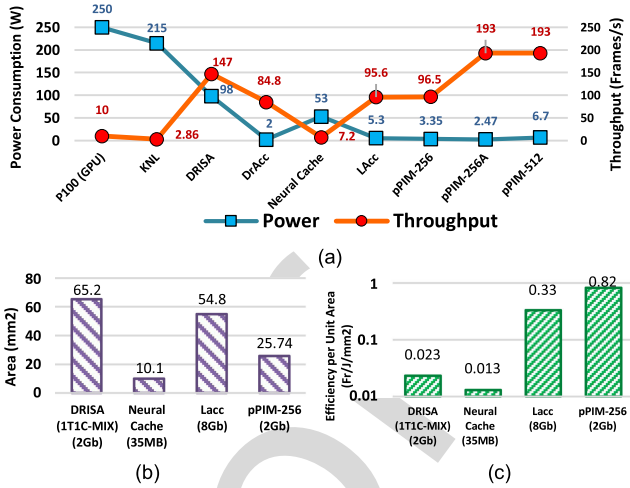

Fig. 3. Comparison of (a) power and throughput of full precision pPIM-256 & precision scaled pPIM-256A with other processors, (b) PIM areas (c) energy efficiency (power/throughput) per unit PIM area with other processors. Area of only PIMs are considered.

Figs. 3b and 3c shows a comparison of area and efficiency per unit area of the PIMs respectively. The pPIM with 256 clusters has a low area overhead (10.64 mm^2) with TG implementation. LAcc [7] reserves 4K lines per bank (of size 16K) for the LUTs and DRISA [5] occupies ~40 percent chip area for implementing the logic, leading to inefficient utilization of memory space. The LUT-based PIMs have higher efficiency per unit area as they leverage memory to implement their functionalities.

3.3 Performance Evaluation With Precision Scaling

We compare the performance and power consumption of the pPIM with full-precision 8-bit operands as well as with precision-scaled 4-bit operands for various CNNs such as AlexNet, ResNet 18,

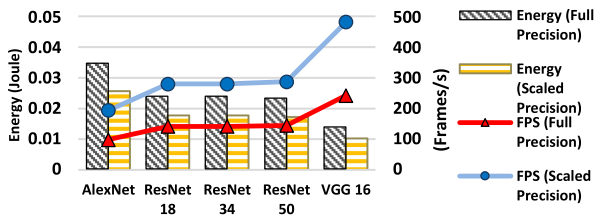

Fig. 4. Comparison of Performance & Energy Consumption of both full precision (8-bit fixed point) and scaled precision (4-bit fixed point) pPIM for different CNN algorithms.

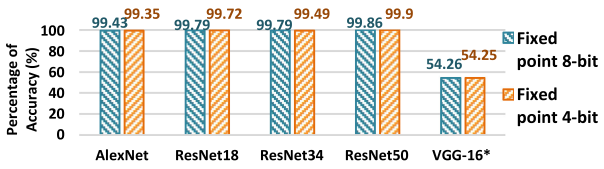

Fig. 5. Accuracy comparison of both full precision (8-bit) and scaled precision (4-bit) pPIM architecture for different CNN algorithms.

ResNet 34, ResNet 50 & VGG 16, in order to investigate the impact of precision-scaled computing on pPIM in Fig. 4. We can see that due to the use of only 4 core-steps with precision scaling, as was shown in Fig. 2, the delay associated with performing one 4-bit MAC operation is half of that for 8-bit MAC, improving its throughput during inference by about 2x. For the 4-bit MAC operation, only 4 single core-steps are required, resulting in approximately 1.35x reduction in power consumption per MAC operation. This enables an overall reduction in energy per frame in inference across all the CNNs as shown in Fig. 4. While precision scaling improves both performance and power, it can result in a loss of accuracy of the CNNs. Corresponding top-5 accuracies on MNIST benchmarks are presented in Fig. 5. For VGG-16 we used a representative set of images from ImageNet for which the accuracy even with 64 floating point precision is only 72 percent. We observe that even with 4-bit fixed point precision scaling, the accuracy of the CNNs do not degrade significantly compared to that in the case of 8-bit fixed point precision. Moreover, the programmability of pPIM enables the user to choose the precision level depending on the application demand.

4 Conclusions and Future Work

In this paper we present the design of a programmable PIM architecture, implemented with LUTs on a DRAM platform that can be programmed to perform versatile CNN operations such as convolutions, pooling & activation function, accompanied by a high-bandwith & low latency data communication model built upon the existing communication infrastructure of DRAM. The performance evaluation of the pPIM chip for multiple CNNs and its comparison against state-of-the-art CPU, GPU and other PIM architectures is presented. We also present the performance and power consumption of pPIM with precision scaling and can observe that it improves throughput and power by 2x and 1.35x respectively over its full-precision operation mode, while sacrificing minimal accuracy. The functional flexibility of the LUT based pPIM enables the expansion of its range of functionality beyond DL applications to finite-element method (FEM) computations, large-scale linear algebraic operations or other scientific applications as well as support for online learning. In future, we intend to implement online learning to take full advantage of the programmability of the pPIM during CNN training.

Acknowledgments

This work was supported in part by the US National Science Foundation (NSF) CAREER Grant CNS-1553264. The authors would like to thank Jack Lombardi & Richard Davis, Air Force Research Laboratories (AFRL), Rome, NY.

References

[1] W. A. Wulf and S. A. McKee,"Hitting the memory wall," *ACM SIGARCH Comput. Archit. News*, vol. 23, no. 1, pp. 20–24, 1995.
[2] K. Ando *et al.*, "BRein memory: A single-chip binary/ternary reconfigurable in-memory deep neural network accelerator achieving 1.4 TOPS at 0.6 W," *IEEE J. Solid-State Circuits*, vol. 53, no. 4, pp. 983–994, Apr. 2018.
[3] V. Seshadri *et al.*, "Ambit: In-memory accelerator for bulk bitwise operations using commodity DRAM technology," in *Proc. 50th Annu. IEEE/ACM Int. Symp. Microarchitecture*, 2017, pp. 273–287.
[4] C. Eckert *et al.*, "Neural cache: Bit-serial in-cache acceleration of deep neural networks," in *Proc. ACM/IEEE 45th Annu. Int. Symp. Comput. Archit.*, 2018, pp. 383–396.
[5] S. Li *et al.*, "DRISA: A DRAM-based reconfigurable in-situ accelerator," in *Proc. 50th Annu. IEEE/ACM Int. Symp. Microarchit.*, 2017, pp. 288–301.
[6] Q. Deng *et al.*, "DrAcc: A DRAM based accelerator for accurate CNN inference," in *Proc. 55th ACM/ESDA/IEEE Des. Autom. Conf.*, 2018, pp. 1–6.
[7] Q. Deng, Y. Zhang, M. Zhang, and J. Yang, "LAcc: Exploiting lookup table-based fast and accurate vector multiplication in DRAM-based CNN accelerator," in *Proc. 56th ACM/IEEE Des. Autom. Conf.*, 2019, pp. 1–6.
[8] D.-I. Jeon, K.-B. Park, and K.-S. Chung, "HMC-MAC: Processing in memory architecture for multiply-accumulate operations with hybrid memory cube," *IEEE Comput. Archit. Lett.*, vol. 17, no. 1, pp. 5–8, Jan 2018.
[9] V. Seshadri *et al.*, "RowClone: Fast and energy-efficient in-DRAM bulk data copy and initialization," in *Proc. 46th Annu. IEEE/ACM Int. Symp. Microarchit.*, 2013, pp. 185–197.
[10] 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 *Proc. IEEE Int. Symp. High Perform. Comput. Archit.*, 2016, pp. 568–580.