3DL-PIM: A Look-up Table oriented Programmable Processing in Memory Architecture based on the 3-D Stacked Memory for Data-Intensive Applications

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

Abstract—Memory-centric computing systems have demonstrated superior performance and efficiency in memory-intensive applications compared to state-of-the-art CPUs and GPUs. 3-D stacked DRAM architectures unlock higher I/O data bandwidth than the traditional 2-D memory architecture and therefore are better suited for incorporating memory-centric processors. However, merely integrating high-precision ALUs in the 3-D stacked memory does not ensure an optimized design since such a design can only achieve a limited utilization of the internal bandwidth of a memory chip and limited operational parallelization. To address this, we propose 3DL-PIM, a 3-D stacked memory-based Processing in Memory (PIM) architecture that locates a plurality of Look-up Table (LUT)-based low-footprint Processing Elements (PE) within the memory banks in order to achieve high parallel computing performance by maximizing data-bandwidth utilization. Instead of relying on the traditional logic-based ALUs, the PEs are formed by clustering a group of programmable LUTs and therefore can be programmed on-the-fly to perform various logic/arithmetic operations. Our simulations show that 3DL-PIM can achieve respectively up to 2.6× higher processing performance at 2.65× higher area efficiency compared to a state-of-the-art 3-D stacked memory-based accelerator.

Index Terms—Processing-in-Memory, Look-up Table, 3-D Memory, Parallel Processing, Data Encryption, Deep Neural Networks

1 Introduction

With the rapid growth of emerging Artificial Intelligence (AI) and data-intensive applications, the memory-centric computing paradigm is quickly gaining attention [1]–[10]. The performance of the traditional computing devices (*i.e.* CPU and GPU) on such memory-intensive applications is limited by the narrow-bandwidth, high-latency off-chip communications with the memory devices of the system [11], [12]. However, processing within the memory chip alleviates this performance bottleneck by exploiting the high internal bandwidth of the memory for parallel computing. At the same time, latency and energy dissipation from data movements are reduced significantly.

The processing-in-memory systems (PIMs) are designed for accelerating applications/tasks with inherent data parallelization such as the AI acceleration tasks that incorporate large-dimension matrix-vector (GEMV) and matrix-matrix (GEMM) multiplications, the cryptography and data encryption tasks [6], etc. Further, the emergence of 3-D stacked memory technologies has inspired the development of high-performance PIM accelerators [7], [13]. This is because of several distinct advantages that the 3-D stacked memory has over the traditional 2-D Dual Inline Memory Module (DIMM). For example, the 3-D stacked memory technologies such as HMC [14] and HBM [15] can achieve up to an order higher bandwidth, and up to 5× higher energy efficiency, and several times larger aggregated die area over the traditional 2-D memory configuration, allowing the placement of more processing circuitry on the same die footprint [16].

However, the memory-centric computing domain has its own challenges that are primarily related to resource limitations and design optimization. These challenges arise when complex and sophisticated processing engines with a large area overhead are incorporated within the small form factor Memory chips [8], [9], [17]. Although this approach provides the functional sophistication required by the applications, it also limits the maximum operational parallelization and low Bandwidth utilization [18]. Despite this, there is a noticeable trend of PIM computing units becoming increasingly larger in the past couple of years [7], [10], [17], [19].

In contrast, shrinking down the ALU footprint would enable the integration of pluralities of ALU within a bank. This would allow each ALU unit to access the within-the-bank bitline-based which is significantly wider than the bank I/O bus. This would unlock the scope for remarkably higher operational parallelism and utilization of data bandwidth. We observe that leveraging look-up tables (LUT) for performing computations [4], [5], [10], [20] is an ingenious way to reduce the footprint of a flexible processing architecture. Unlike a CMOS-logic-based ALU in which different logic blocks are dedicated to specific operations, a single LUT can perform various operations depending on how it has been programmed. Therefore, a LUT can execute many

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

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

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

different logic/arithmetic functions without incurring incremental area, power, and latency overheads for each functionality [4], [5], [10], [20]–[22]. However, LUTs should only be utilized for low-precision computing (*e.g.* 4-bit) [10] since the area of a LUT scales up exponentiallyc with the operand precision and may occupy a large area for large operational precision.

Conveniently, the usage of excessively large LUTs can be averted by decomposing a large-precision computation into a series of lower-precision sub-operations and executing those via a group of tiny LUTs instead. By assembling such a group of LUTs together, it is possible to perform larger-precision, complex operations with minimal LUT overheads. However, these LUTs must also be programmable (*i.e.* re-writable) such that these can be adapted for performing a wide range of logic/arithmetic operations required to support a diverse range of applications. **To this end, we propose a 3-D stacked memory-oriented PIM architecture (3DL-PIM) consisting of many low-footprint, programmable PEs formed of interconnected LUT clusters. By leveraging the programmability of the constituent LUTs, the proposed PEs can diverse domains of applications, such as AI acceleration, Data Encryption, and Matrix-Matrix/Vector arithmetic. We also integrate a multitude of these PEs within the memory banks to expose a collectively wider data bandwidth to the PEs and achieve improved operational parallelism.** With the aid of a custom, programmable Instruction Set Architecture (ISA), the proposed PIM can be programmed (*i.e.* re-written) on the fly to support a wide range of logic/arithmetic operations for performing versatile tasks *e.g.* data encryption, linear algebraic applications, and AI training and inferences.

For maximum performance optimization, it is also essential to facilitate fast communication among the PEs. While prior works have relied on data-copying between banks [8]–[10] or have used complex and expensive networks [23]–[25], we instead leverage a low-cost bitline-based internal communication mechanism [26] to facilitate high-bandwidth and low-latency communication channels among groups of PEs within the banks. We evaluate the performance of the proposed 3DL-PIM architecture for multiple application domains, including Deep Neural Network (DNN) acceleration, Data encryption, and Matrix-Vector Arithmetic. We also compare the performance with state-of-the-art traditional processors such as CPUs, GPUs, and other contemporary 2-D and 3-D memory-based accelerators. To summarize, in this work we propose the following novel contributions:

1) We propose 3DL-PIM, a 3-D stacked memory-based memory-centric processing architecture with many programmable, tiny, LUT-based PEs for flexible and energy-efficient data-parallel computing. In order to implement a specific logic/arithmetic operation, these PEs are programmed appropriately, making it possible to implement an arbitrary number of different logic/arithmetic operations without any incremental overhead. The PEs are incorporated within the memory banks for optimal usage of the internal data bandwidth of the memory organization.
2) We propose a custom bank-level controller, accompanied by a custom, programmable Instruction Set Architecture (ISA) for programming the LUTs to perform specific logic/arithmetic operations and also for conducting the execution of the operation via fixed-length instructions.
3) We evaluate the proposed architecture for a wide range of applications including Data Encryption, Linear Algebraic Applications (BLAS), and AI Acceleration (*i.e.* CNNs and Transformer).

The remaining part of the paper is outlined as follows: **Section 2** provides a brief overview of the related works, **Section 3** discusses the microarchitecture of 3DL-PIM in detail, **Section 4** demonstrates the LUT-based implementations of important logic/arithmetic operations in the 3DL-PIM PEs, and **Section 5** discusses the architecture and the operational mechanism of the 3DL-PIM bank Controller Unit (BCU) and Instruction Set Architecture (ISA). These are followed by **Sections 6** covering various technical aspects of the hardware-level integration of the PEs within the memory organization. Finally, **Section 7** presents the experimental evaluations of 3DL-PIM and its comparative benchmark with the state-of-the-art, which is followed by concluding remarks in **Section 8**.

2 Related Works

The 3-D memory architecture is formed by stacking multiple DRAM dies vertically on top of a base logic die [14]. The stacked dies are interconnected with a large number of vertical through silicon vias (TSV). The 3-D stacked memory technologies gained popularity as a potential solution to the 'memory wall' problem, thanks to its TSV-based I/O bandwidth which can be several times higher than the standard 2-D DRAM chips [16]. Two distinct 3-D stacked memory organizations have been proposed so far: a) Hybrid Memory Cube (HMC), and b) High Bandwidth Memory (HBM/HBM2). The organization of HMC is partitioned into many vertical columns called 'vaults' that consist of vertically aligned segments of each stacked die and its localized TSV bundle. On the other hand, HBM features centralized TSV bundles and a memory organization closely resembling the standard 2-D DRAM. The TSV interconnects act as channel I/Os that are interfaced to planar bank-group buses in each die.

The 3-D stacked memory-centric accelerators can be classified either as Near Memory Accelerators, with the processing logic located only on the base die [23], [24], [27], or Near Bank Accelerators, with processing logic located within the memory dies [7]–[10], [17], [28], [29]. In general, the Near-Bank Accelerators enjoy lower data-access latencies and improved data bandwidth than the Near Memory Accelerators, albeit at a higher design complexity. Some Near-Bank Accelerators, such as SpaceA [29] incorporate heterogeneous processing logic on different stacked memory dies to facilitate a streamlined execution flow in the vertical direction. These accelerators may be aimed at a single specific application [10], [17], [23], [24], [29], or capable of supporting multiple application domain/ general purpose computing [8], [9], [28].

While the Near-Bank Accelerators [8]–[10], [17], [28], [29] offer better optimization of data access/ communication overheads compared to the Near-Memory Accelerators [23], [24], [27], [30], these are still able to only utilize a fraction of the bank-internal bandwidth of DRAM. Therefore, a significantly higher data-bandwidth utilization would be possible if the processing logic were to be placed within the banks. However, the Processing Engines in the existing Near-Bank Accelerators are far too large to be fitted within the banks [8]–[10], [17]. This inspires us to develop compact, lightweight Processing Engines/Elements that can be placed within the banks to expose significantly wider, low-latency datapath between the memory and the processor.

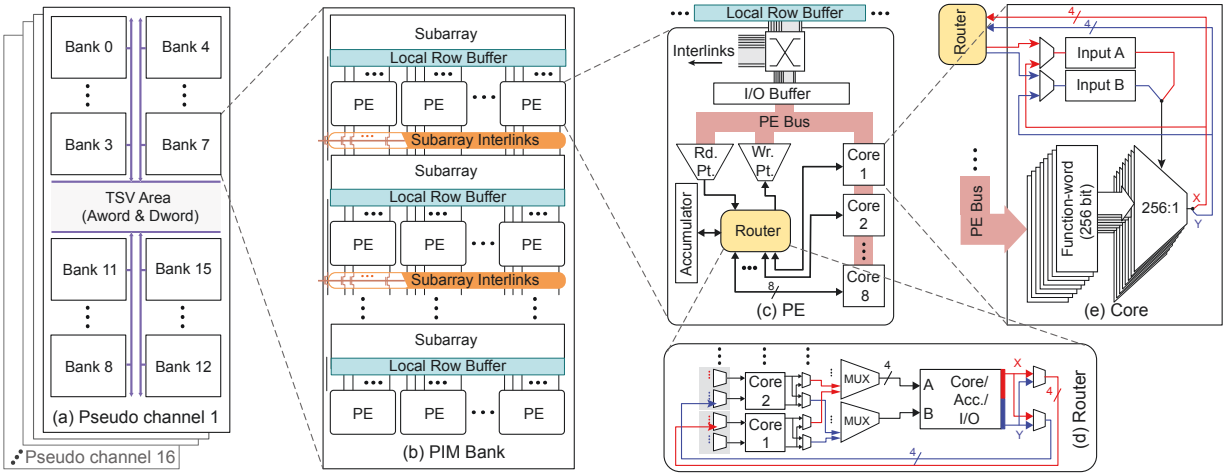


Fig. 1. Overview of the 3DL-PIM architecture integrated, including a) an overview of the organization of the used memory platform (HBM2 in the Pseudo Channel mode), b) arrangement of the 3DL-PIM PEs inside a memory bank, c) architecture of a PE, d) a detailed view of the router of a PE, and e) a detailed view of a LUT-core in a PE.

3 Proposed 3DL-PIM Architecture

3.1 PIM Microarchitecture

3.1.1 Overview

The proposed 3DL-PIM architecture consists of many parallel processing elements placed within the memory banks. A DRAM bank is a 2-D array of memory cells. Each row of data in this array is connected to a common *wordline* which can be activated via a row address. Once activated, a *page of data* is released on to the *bitlines* which are sensed and latched by the sense amplifiers (SA). One of several *columns* of data can be selected from the sense amplifiers via the column address. The physical organization of a DRAM bank, however, is more complex than that. Inside a bank, groups of wordlines are bundled together to form *subarrays* [31], primarily with the goal of reducing the physical lengths of the bitlines for improving electric and capacitive properties. Each subarray has its own set of SAs and local row buffers that copy data from the SAs. The proposed 3DL-PIM PEs are located in rows between each pair of subarrays and are interfaced with the local row buffers of the respective subarrays via extended bitlines.

Our approach of integrating the PEs within the memory banks makes 3DL-PIM compatible with different 3-D stacked memories including HMC [14], HBM [15], and HBM2 [32]. In this section, we discuss the implementation of this architecture on the HBM2 memory platform which consists of eight channels distributed across eight channels. Each channel is further divided into a pair of pseudo channels (pCH) consisting of sixteen banks. The 128-bit channel I/O bus is split into a pair of 64-bit wide pCH I/O buses that can operate autonomously in pCH mode of operation. Figure 1 presents a hierarchical view of the 3DL-PIM architecture, showing a) a higher-level memory organization of HBM2, b) the arrangement of the PEs inside a PIM-bank, c) the architecture of a PE, d) the architecture of the router of a PE, and e) the architecture of a LUT-core.

3.1.2 PE Organization

Figure 1(a) and (b) respectively shows the baseline memory organization and the arrangement of the 3DL-PIM PEs inside a bank. We refer to the banks containing the PEs as the 'PIM-Banks'. The PEs are arranged in horizontal arrays (rows) inside the PIM bank such that each row of PE is located between a pair of subarrays. The placement of the PEs inside the bank is facilitated by replacing a portion of the subarrays [7]. Each PE is able to directly interact with the subarray above it via the subarray's local row buffer. Four PEs in a row have shared access to any subarray. The row of PEs can also communicate with other subarrays via a set of subarray interlinks [26], as discussed later in Section 3.1.6.

3.1.3 PE Microarchitecture

3DL-PIM PEs are designed to perform a wide range of logic/arithmetic operations exclusively using LUTs. The LUT-based computing approach achieves improved energy efficiency and lower latency than the traditional ALU-based computing paradigm due to having comparatively fewer stages of transistors and fewer logic-switching activities per operation [4], [5]. Each PE consists of eight Look-up Tables (LUT) based processing cores, termed the 'LUT-cores', for performing micro-computations, as shown in Figure 1(c). A LUT-core contains one 8-bit LUT that can be programmed to perform any logic/arithmetic operation on a pair of 4-bit operands or a single 8-bit operand. The LUT-cores as well as the other components of the PE are interconnected by a router that facilitates parallel communications among all the interconnected nodes. With the aid of the router, the PE can combine parallel operations across all the LUT-cores in multiple stages to implement more complex and sophisticated operations.

As shown in Figure 1(c), alongside the LUT-cores and the router, a PE also contains a 16-bit accumulator register, an internal Bus, and an I/O buffer for managing the data flow in and out of the PE. The I/O buffer accesses and latches data in large granularity from the local row buffer of the neighboring memory subarray via a crossbar switch. The data buffering enables the PEs to maximize data re-use, especially for applications with irregular memory access patterns. The I/O buffer is conveniently sized to 512-bit so as to achieve a balance between buffering capacity and circuit overhead.

3.1.4 LUT-core Microarchitecture

The LUT-Cores act as 8-bit processors with programmable functionality. Figure 1 (e) shows a detailed view of the LUT-core architecture. Each LUT-core contains an 8-bit LUT formed of eight 256-bit buffers, paired with an 8-bit 256:1 Multiplexer. The LUT is programmed using a set of eight 256-bit *function-words*. A LUT-core additionally contains a pair of 4-bit Input registers, A and B that receive the operands from the router. These registers collectively control the select pins of the LUT to perform a LUT 'look up' and access a pre-programmed output. The output of the LUT-core is either forwarded to the router or recirculated to the input registers for repetitive operations.

3.1.5 Intra-PE Communication

A PE contains an internal Bus of the same width as the I/O buffer that allows the content of the I/O buffer to be distributed to different components of the PE, as depicted in Figure 1(c). The I/O buffer may contain a batch of up to sixty-four 8-bit data operands called the *data-words*, or a pair of 256-bit *function-words*. The LUT-cores are able to read the *function-words* directly from the PE-bus. On the other hand, the *data-word* are accessed from the I/O buffer in pairs by the router via the Bus. A Read/Write Pointer enables the router to select a pair of data-words at a time. Once received by the router, the data-word pair is distributed to the LUT-cores that perform a series of same or different operations.

The microarchitecture of the PE's router is shown in detail in Figure 1(d). It is essentially a crossbar switch, implemented with a set of 4-bit multiplexers. The router is capable of forwarding the 8-bit output of a LUT-core either to the input port of another LUT-core, the PE accumulator, or the I/O buffer via the PE bus. Also, it can perform this task in parallel across all the LUT-cores. Each of the communication channels of the router has a pair of 4-bit sub-channels that are parallel and independent of each other. Therefore, an 8-bit output of one LUT-core can be fed to the same or two different LUT-cores as two individual 4-bit operands.

3.1.6 Inter-PE Communications

Although the PEs can directly access only their neighboring subarrays, they also have passive access to all the subarrays in the PIM bank. This is facilitated by leveraging a subarray-interlinking mechanism that connects the local bitlines of every neighboring subarray via access transistors [26]. As shown in Figure 1(b), this mechanism enables a page of data to hop across multiple subarrays at a time via the interlinks to reach a different destination subarray and be accessed by the adjacent row of PEs.

3.2 Data Access Mechanism

The PEs access data from their neighboring subarrays via the local row buffer of the subarray. First, a target wordline is activated which causes a page of data to be copied from the corresponding memory row to the local row buffer. In the proposed configuration, four PEs are connected to the local row buffer such that a 2kb wide segment of the buffer is visible to each PE. A PE accesses this data via multiple read operations. Since the I/O buffer of the PE has a capacity of 512b (64B), the PE needs to perform exactly four consecutive reads. A crossbar switch allows the PE to access a different 64B block of data during each read, as depicted in Figure 1(c). Conversely, the outputs are written back to the local row buffer from the I/O buffer of the PE via the crossbar switch.

3.3 Control Architecture

3.3.1 Bank-level Control

The 3DL-PIM architecture is equipped with multiple custom-designed controller units to operate the PEs. All the PEs in a PIM-bank are controlled by the same Bank Controller Unit (BCU). Further, the corresponding PIM-banks across all the paths share a common BCU. Figure 2 (a) shows the connectivity scheme of the BCUs to a set of PIM-banks. The BCU has a custom-designed Instruction Set Architecture (ISA) that decodes a stream of instructions generated by a host device and outputs corresponding control-words for operating all the PEs inside a PIM-bank. The BCU has a custom-designed architecture specifically aimed to operate the PEs. It leverages the control-TSVs to transmit 128-bit wide PE control-word per clock to operate the PEs in the designated memory banks. The detailed operation of the BCU is discussed in detail in Section 5. Using a single BCU for controlling all the PEs in a bank inevitably enforces operational homogeneity within a Process Column. However, since the targetted applications are characterized by highly regular and repetitive operations (*e.g.* MAC), this does not tend to be a significant performance limiting factor.

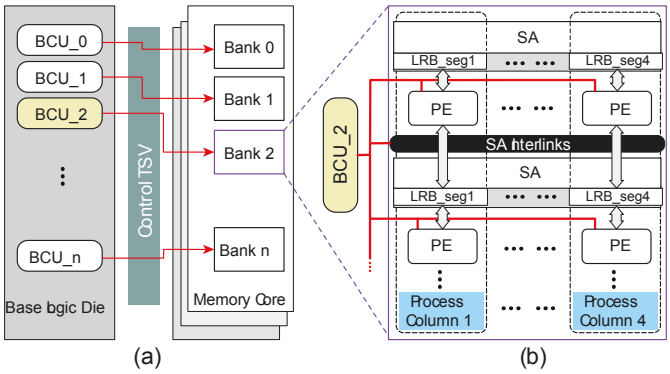


Fig. 2. The overview of the 3DL-PIM control architecture, includes a) a higher-level view, and b) a bank-level view. The PIM-banks across the Pseudo Channels (pCHs) in the Memory Cores presented as Bank 0, 1, 2, $\cdots$, n, are operated by Bank Controller units (BCU) placed in the logic die. Inside each PIM-bank, multiple parallel Process Columns are operated synchronously by the corresponding BCU.

The BCUs are placed in the logic die of the 3-D stacked memory and utilize the control-TSVs to transmit the control signals to the target bank across multiple paths. The placement of the BCUs in the logic die is inspired by two factors. First, the BCU features a relatively larger amount of logic circuitry, including registers, counters, decoders, and multiplexers. Second, since each BCU operates multiple banks across all the stacked memory chips, it is easier to distribute control signals from the base die via the TSVs.

3.3.2 Process Columns

Inside each PIM-bank, the PEs are arranged in rows between pairs of subarrays. The placement of the PEs is facilitated by replacing several subarrays in the bank. As discussed previously in Section 3.1.5, novel 'subarray interlinks' are leveraged to bridge the local bitlines of the neighboring subarrays [26]. This communication essentially leads to the forming of a novel multiprocessing architecture called the **Process Column**, consisting of the PEs aligned along a column (Y-direction), as shown in Figure 2(b). A Process Column also includes the portions of the subarrays that the corresponding PEs are connected to. Each bank contains four Process Columns that are operated by the same BCU, effectively forming a 4-way parallel Single Instruction Multiple DATA (SIMD) processing layout. The subarray segments belonging to a particular Process Column act as shared memory for all the PEs in that Process Column. As a result, these PEs can share the workload of a common task, *e.g.* the workload from a Neural Network Layer. Contrary to the widely used 2-D spatial arrangements (*e.g.* the 2-D systolic arrays implemented in the Nvidia Tensor Cores and Google TPU), Process Column is essentially a 1-D multiprocessing layout aimed at simplifying the control architecture and minimizing circuit overhead. While this approach requires the mapping of Matrices and Vectors to be flattened to a 1-D spatial distribution, this does not compromise the system's performance in any way. However, the added benefit of utilizing the Process Column layout is a significant reduction in

the complexity of the spatial data/operation distribution compared to an equivalent 2-D multiprocessing layout.

4 Implementation of Operations

Each PE of the proposed 3DL-PIM architecture can perform various complex operations by executing simpler, lower-precision computations in the LUT-cores in multiple stages. By programming the LUT in the core in a specific way, it can be used to perform virtually any operation such as bitwise logic, addition, subtraction, multiplication, shifting, substitution, comparison, pooling, etc. As a result, it is compatible with multiple application domains, including AI-oriented applications such as CNN, RNN, LSTM, and Transformer acceleration, linear algebraic applications such as matrix/vector arithmetic, finite-element methods, and cryptographic applications such as AES encryption.

4.1 Multiply-and-Accumulate (MAC)

The MAC operation is the building block for any matrix-vector (GEMV) and matrix-matrix (GEMM) multiplication task and therefore is extensively utilized by fully connected, convolutional, and attention layers in the DNNs. Conventionally, a MAC operation is performed via sequential execution of individual multiplication (MUL) and addition (ADD) operations. However, we partially parallelize these two operations and thereby combine them into one continuous operation that requires fewer clock cycles to execute. Alongside minimizing the latency, this also maximizes resource utilization in the PE. The 8-bit MAC operation is decomposed into a series of 4-bit multiplications and 4-bit additions in multiple clock steps. First, both of the 8-bit operands, a and b are disintegrated into 4-bit segments: a_H-a_L, and b_H-b_L. Then these segments cross-multiplied to generate four partial products V_0-V_3 such that V_0=$a_L \times b_L$, V_1=$a_L \times b_H$, V_2=$a_H \times b_L$, and V_3=$a_H \times b_H$.

Next, the partial products are added in several stages to generate the product of a and b. Also, the product is added alongside the accumulation from the previous rounds of MAC operations using the scheme shown in Figure 3(a). Figure 3(b) shows the programming scheme of the LUT-cores in a PE for performing this operation. Three LUT-cores in the PE are programmed as 4-bit multipliers and five LUT-cores as 4-bit adders. Finally, the step-wise implementation of the whole process inside a PE is presented in Figure 3(c).

As can be seen from Figure 3(c), the accumulation is initiated at t=2, as soon as the partial products V0-V3 are generated. The 16-bit accumulated MAC output of two 8-bit inputs is generated as four 4-bit segments A0-A3 and is contained in the accumulator. During different stages of the accumulation operation, the 4-bit segments of the previous MAC output are fetched from the accumulator. These values are represented as A0-A3 written in green text in Figure 3c. The new MAC output overwrites the older values in the accumulator. It takes only eight clock cycles to perform the whole operation.

4.2 Maxpooling

The maxpooling operation is primarily required by CNNs for downsampling the feature maps. In order to perform 4-bit max-pooling, a LUT-core can be programmed to compare between a pair of 4-bit inputs and forward the comparatively larger value as the output. By cascading two such 4-bit maxpooling LUT-cores, it is possible to implement an 8-bit maxpooling operation. The 8-bit Max-pooling requires only three LUT-cores and therefore can be executed two-way parallel in a PE. The Maxpooling layer,

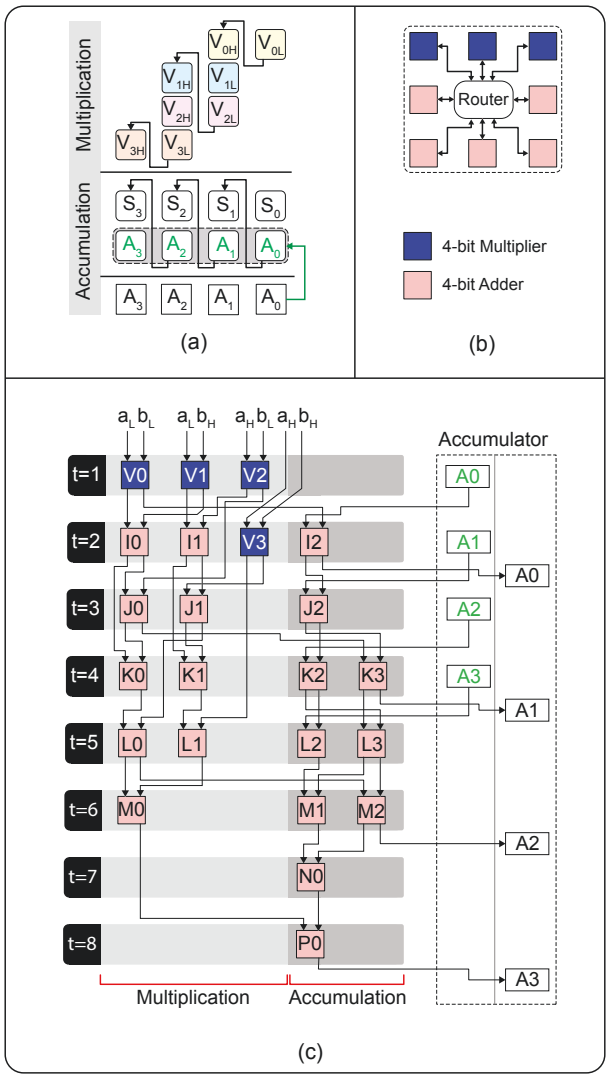


Fig. 3. Overview of 8-bit fixed-Point multiply-and-accumulate (MAC) operation in 3DL-PIM. (a) shows the operation-decomposition algorithm utilized for representing the 8-bit in terms of 4-bit operation, (b) shows the programming scheme of the LUT-cores of a PE for performing 8-bit MAC, and (c) shows the step-wise implementation of the operation inside a PE. In this figure, the left and right arrows coming out of each LUT-core represent the most and least significant 4-bit of the outputs of the LUT-cores respectively. Green texts represent data from the prior round of operation. The clock steps of the operation are designated by the values of 't' while the letters I, J, K, L, M, N, and P respectively represent the corresponding stages of the partial addition operation. During each clock, the numeric tags accompanying the letters designate the concurrent operations across the LUT-cores.

which is also implemented by sliding a filter across an activation map, can be orchestrated in a similar manner to the Convolutional Layer. This involves multicasting the Maxpooling filter, followed by the distribution of even slices of the activation map to each PE in the Process Column.

4.3 Substitution/Transformation

Substitution operations are required by multiple application domains. For example, ReLU and TanH Activations, required by the DNN acceleration applications can be performed via LUT-substitution. Similarly, Rijndael S-BOX substitution and Galois Field multiplication for AES Encryption can be carried out by replacing the input with a predefined value as output. Therefore, these operations can be performed in a single step in a LUT-core.

4.4 Other Operations

Alongside the aforementioned operations, the proposed 3DL-PIM PE can also support standard bitwise logic operations as well as arithmetic operations such as addition, subtraction, modulus, etc. Other operations involve bit-rotation, left and right shifting, and bit-padding operations. Further, 3DL-PIM can also support all the aforementioned operations with other fixed-point precision of the data operands (*i.e.* 6-bit/4-bit), with improved performance and energy efficiency. In Section 7 we present performance evaluation for AI inferences with 4-bit precision. This feature also opens up opportunities to implement various approximate computing techniques on the same hardware.

5 Instruction Set Architecture

In order to implement different logic/arithmetic operations discussed in Section 4, the 3DL-PIM architecture is equipped with custom-designed Controller Units. Identified as the Bank Controller units (BCU), each unit is responsible for operating all the Processing Elements (PE) within a bank. The BCUs are placed in the logic die of the 3-D stacked memory and utilize the control-TSVs to transmit the control signals to the banks.

5.1 Bank Controller Architecture

5.1.1 Instruction Format

Figure 4 shows a high-level view of the ISA of the BCU. The BCU consists of an Instruction Register, an Instruction Decoder, several internal registers, Counters, a Controller/Sequencer unit, and Data and control buses. It utilizes a fixed-length instruction-word format that can be divided into three segments: 5-bit wide **Opcode**, 16-bit wide **PE Mask**, a 10-bit wide **Pointer**. The **Opcode** represents various operations supported by the 3DL-PIM ISA. In the current format, the ISA can support up to 32 different Operations. Table 2 lists several fundamental Instruction Opcodes and their functionalities.

The **PE Mask** bits of instructions provides selectivity over which PEs will execute a SIMD instruction. By setting a Mask bit to high, the corresponding PE is excluded from executing the instruction. The **Pointer** bits of an instruction-word allows a fine-grained selection of operands in the I/O buffer. The **Pointer** bits may a) hold the row and column address of a block of data in the subarray, b) act as a pointer to *function-word* a LUT-core latches, or c) act as data-pointer to the I/O buffer of the PE.

5.1.2 Controller/Sequencer

The **Opcode** is decoded in the Instruction Decoder and forwarded to the Controller/Sequencer unit, as shown in Figure 4. An opcode points to the first control-word corresponding to the instruction inside the Controller/Sequencer unit. The following control-words for that instruction are executed sequentially during the consecutive clock cycles. The control-words are distributed across the ISA as well as to all the PEs via the control bus. As indicated in Figure 4, a control-word has two distinct segments. One segment consists of control signals for the PE, identified as the PE control-word. The other segment controls the internal registers of the BCU itself, identified as the BCU Control. PE control-word consists of router control commands and PE data flow control commands. A detailed discussion of the PE Control mechanism is presented in Section **3.3.2**.

TABLE 1
Specifications of various fixed-Point Operations in a PE

Operation	# of Clocks	Parallel Ops./PE
8-bit MAC	8	1
8-bit Maxpool	4	2
8-bit Substitution	1	8
4-bit MAC	5	4
4-bit Maxpool	1	8
4-bit Substitution	1	8
4-bit Bitwise Op	1	8
8-bit Bitwise Op	1	4
16-bit Bitwise Op	1	2

The Controller/Sequencer unit is micro-coded on a programmable memory device. This design choice is inspired by multiple advantages. First, micro-coding of the control-words reduces the use of logic circuitry and simplifies the designs. This also reduces the power consumption compared to an equivalent logic-based Controller. Second, new operations can be implemented or the existing operations can be modified/optimized in the future by simply reprogramming the control-word bits.

5.1.3 Implementation of the Instructions

The Instructions supported by the BCU are either memory access instructions, logic/arithmetic instructions, or LUT programming instructions, as shown in Table 2. Depending on the nature of the instruction, the Pointer bits of an instruction-word may be used as a memory address, a data pointer in the I/O buffer, or a LUT pointer. For **LDB** and **STB**, which are memory access instructions, the Pointer bits represent the address to a particular column in a DRAM row. For logic/arithmetic instructions such as **MAC, RLU,** and **MPL**, the Pointer bits of an instruction act as a pointer to data in the I/O buffer, which can contain up to 64 8-bit data/ 128 4-bit data operands. The **PRG** instruction reprograms the LUT-cores of a PE in real-time using new *function-words* and leverages the Pointer bits as LUT addresses. Each of the eight LUT-cores in a PE contains eight 256-bit latch banks and therefore the Pointer bits can select any of the sixty-four latch arrays in a PE for reprogramming.

5.2 PE Control Mechanism

A PE is operated by the 128-bit wide PE control-word received via the control-TSVs, as shown previously in Figure 4, and in more detail in Figure 5. This includes Routing signals as well as PE Dataflow control signals. The routing signals control the router of the PE to facilitate parallel communications among all the components of the PE, each of which is regarded as a communication node. A PE contains twelve communication nodes, including eight LUT-cores, two I/O ports, and an accumulator register with two ports, as shown in previously Figure 1(c) and 1(d). Each node has a pair of 4-bit independent outlet channels, X and Y, that are respectively operated by the **X Route** and **Y Route** signals. Both channels can transmit data to twelve different destination nodes (including their own node). Furthermore, the data can switch over between X and Y channels at each node via a set of **Switch** signals. The **X Route**, **Y Route**, and **Switch** signals for all twelve nodes make up the router control segment of a PE control-word. As seen in Figure 5, the PE Dataflow control signal consists of two 10-bit segments called **A_En** and **B_En** that are used for activating/deactivating ten pairs of internal registers.

6 Design Choices and Motivations

In this section, we critically discuss some of the key design choices made during the development of the proposed 3DL-PIM architecture and justify the motivations.

LUT-Cluster-Based Computing: One key challenge of performing computations using LUTs is that the size of a LUT scales up exponentially with the bit-precision of the operands [4]. We overcome this challenge by replacing large LUTs with a group of smaller, 8-bit LUTs within each PE. The collective footprint of

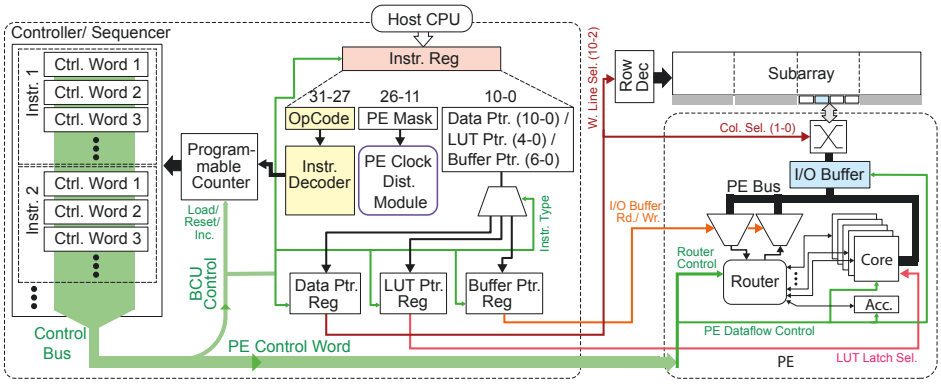


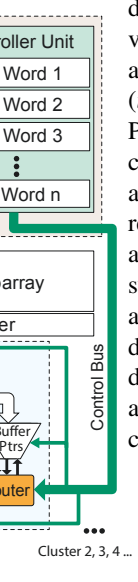
Fig. 4. Instruction Set Architecture of the Bank Controller Unit (BCU) of 3DL-PIM. A 32-bit wide fixed-length instruction is decoded to generate control-words from a micro-coded Controller/Sequencer unit. A control-word is distributed across the internal registers as well as the PEs, via the control bus and the control-TSVs.

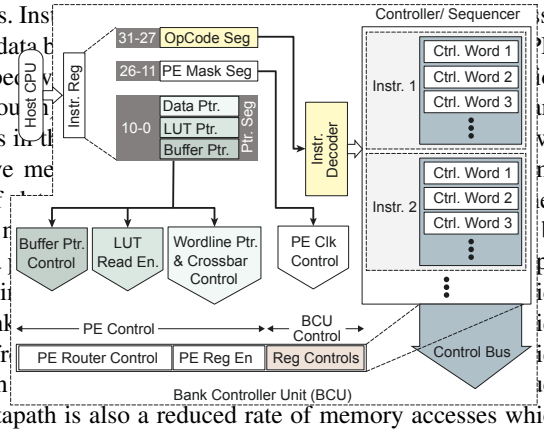
TABLE 2
CNN-Oriented Instructions and their functionalities

Opcode	Functionality
LDB	Reads the contents of a memory row in the I/O buffer
OUT	Writes the PE accumulator back to a specific location in the I/O buffer and resets the accumulator
STB	Writes I/O buffer back in the designated Memory Row
PRG	Programs a LUT-core with the contents of the I/O buffer
MAC	Multiply and Accumulate two 8-bit data from the I/O buffer
RLU	ReLU Activation on multiple 8-bit data from the I/O buffer
MPL	Max-pooling out of a pair of 8-bit data from the I/O buffer

such a group of LUTs is multiple orders lower than a single, larger LUT. For example, replacing a 16-bit LUT-core with a group of eight 8-bit LUT cores results in a 64× reduction of area overhead. However, in order to implement the same set of operations using smaller LUTs, various operation decomposition techniques are to be leveraged [4], [10]. Therefore, we form the 3DL-PIM PE by combining a group of 8-bit LUT-Cores and interconnecting these cores with a flexible router that allows seamless distribution of decomposed micro-operation among the LUT-cores within the PE.

In-Bank Computing: We place the PEs within the memory banks so that these are not dependent on the narrow-width bank I/O for data access. Ins s a very high data b Es are equippe ion (*i.e.* one-fou nk PE such as i wo consecutive me me amount of eir respective r by adopting a ple subarrays i ide a PIM-bank ide datapath fr ide datapath in ch a wide datapath is also a reduced rate of memory accesses which contribute to lowering power consumption.





Alongside this, the interlinked bitlines of the subarrays make it possible to transmit the same amount of data concurrently to all the PEs inside the Process Columns. Overall, 3DL-PIM PEs enjoy a remarkably superior bandwidth than the inter-PE communication based on the bank I/O bus [7], [10] or custom networks [25], [33] proposed by prior 3-D stacked memory-based PIMs. Furthermore, the PEs in the Process Column have shared access to all subarrays. Therefore, the subarrays collectively act as a large shared cache to the Process Column and facilitate optimum data re-use.

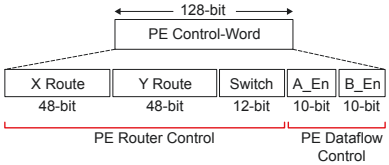


Fig. 5. A detailed view of the PE control-word received by the PEs via the control-TSVs.

Our performance evaluations, presented in Section 7, include the latency and power consumption associated with the in-bank data accesses and data movements. While the data movement latencies are modeled using the timing parameters of the interlinked subarrays [26], the data access latencies are based on the standard DRAM timing parameters such as the row access latency, tRAS). This is because, although the proposed model of incorporating PEs within a bank involves the replacement of several subarrays, the functionality of the remaining ones is unaffected by this modification. Further, the PEs are interfaced to the sense amplifiers on the bitlines, which effectively prevents the PEs from loading the bitlines, unlike prior bitwise processing architectures [34], [35].

Power Optimizations: We choose to implement the proposed 3DL-PIM architecture on 3-D stacked memory (*e.g.* HBM/HBM2) because of improved power supply and a larger aggregated chip area than the 2-D DIMM DRAM configuration [36]–[39]. Moreover, the physical I/O pins, which account for a large portion of power consumption (*e.g.*up to 80% for HBM2), remain inactive during PIM operation [7], [8], [13]. Alongside, the PEs of the proposed 3DL-PIM perform internal buffering of blocks of data in order to reduce the rate of memory activity rate significantly, as discussed previously in Section 3. Therefore although we integrate a large number of low-power, LUT-based PEs in each stacked HBM2 memory chip, the overall system power consumption of 3DL-PIM remains near the rated power of the standard memory module (*i.e.* 10W/chip) [13]. The power consumption figures of different 3DL-PIM configurations are reported in Table 4.

7 Experimental Evaluation

7.1 Experimentation Setup

The PEs and the BCU of 3DL-PIM are designed and verified in the HDL environment. The device parameters are reported in the 20nm technology node and are obtained from ASIC physical synthesis of the PEs using the Synopsys Design Compiler tools. The synthesis is performed in compliance with memory chip specifications, *e.g.* only four metal layers are utilized in the synthesis of the PEs. We also present the inter-subarray communication parameters in [26] the 20nm technology node. For modeling the

latency and energy consumption from data accesses (read/write) by PEs from neighboring subarrays, we utilize the tRCD (Row-to-Column Delay) timing and the read/write energy for DDR4 DRAM in the 22nm technology node [40]. The hardware performance parameters are summarized in Table 3.

We utilize a Python-based analytical model of the 3DL-PIM PE to formulate different logic/arithmetic operations in the LUT-based processing environment of a PE. These operations are then implemented in the HDL simulation model for further verification.

7.2 Device Configurations

We evaluate the proposed 3DL-PIM architecture via its HBM2-based implementation. HBM2 is a state-of-the-art 3-D stacked memory architecture that has been explored for PIM adoption [7]. We integrate 3DL-PIM in the memory by re-purposing multiple banks in each pCH. Each PIM-bank contains four Process Columns, operated by a shared BCU. We present three device configurations of 3DL-PIM: **LP-3DL-PIM**, **MP-3DL-PIM**, and **HP-3DL-PIM**, with different number of PIM-banks in each pCH. Table 4 presents different technical aspects of the three configurations. The purpose of adopting three different configurations is to offer adaptability to various application domains with varying computational demands and power specifications. For example, the LP-3DL-PIM configurations are aimed at the low-power edge and mobile applications with power consumption below the rated power (*i.e.* 30W) of the standard HBM2 module [38]. On the other hand, HP-3DL-PIM is a high-performance variant of 3DL-PIM with 4× higher parallel processing performance than the LP-3DL-PIM configuration.

7.3 Performance Benchmarking

7.3.1 AES Encryption

We choose to implement the Advanced Encryption Standard (AES) encryption algorithm to demonstrate the capability of the proposed 3DL-PIM architecture to perform data-encryption. Besides the AES encryption algorithm is characterized by LUT-friendly computations [22]. AES is a fixed-size block-cipher-based encryption algorithm. Each data block (*i.e.* plaintext) undergoes several iterations of an encrypting function called the *Round Function*, consisting of four consecutive processes of *Subbytes*, *Shiftrows*, *Mixcolumns*, and *Add Roundkeys*. The proposed 3DL-PIM architecture can perform all the logic/arithmetic operations required for performing AES Encryption(*i.e.* s-box substitution, bitwise XOR, Galois-field multiplications, etc.), along with in-situ Key Expansion for 128/192/256-bit keys. We evaluate AES Encryption performance for the 128-bit key which involves fourteen iterations of the Round Function. Each PE spends 1223 clock cycles to process one plaintext block, accompanied by 440 clock cycles of operation for performing the corresponding key expansion. The AES encryption throughput is evaluated for HP, MP, and LP-3DL-PIM configurations and compared with the Nvidia GTX TITAN X (Maxwell) GPU in Figure 6(a). Even though AES encryption is a compute-bound application, all configurations of 3DL-PIM achieve higher processing throughput than Titan X. This is primarily due to having a significantly larger number of compact PEs working in parallel than the GPU for all cases. Moreover, 3DL-PIM also achieves 22.9× higher energy efficiency in this application by: a) nearly eliminating the energy overhead from data movements, and b) performing computations on the proposed LUT-based PEs instead of the traditional logic-based ALUs.

TABLE 3
RTL Properties of 3DL-PIM components

Component	Latency (nS)	Max. Clock Freq. (GHz)	Power (mW)/ *Energy(μJ)*	Area (mm^2)
LUT-core	0.63	1.59	0.93 (Dyn.) 0.45 (Leak.)	0.002
Processing Element (PE)	0.63	1.59	6.24 (8-bit) 5.43 (4-bit)	0.021
Bank Control Unit (BCU)	0.392	1.59	0.079	0.0005
Inter-subarray Comm. (1/7/15 hops)	106.07/ 140.35/ 186.07	N/A	*0.033/ 0.043/ 0.062*	N/A
Data Access (tRCD)	6.77	N/A	*0.0011*	N/A
TSV	1	2.133	4.6 x 10^{-6}	−−−

TABLE 4
Hardware Specifications of the 3DL-PIM Configurations

Features	LP-3DL-PIM	MP-3DL-PIM	HP-3DL-PIM
Total PIM-banks	32	64	128
P. Column/ Pseudo channel	8	16	32
No. of BCUs	2	4	8
PIM Circuit Overhead	9.53 %	19 %	38.1 %
Max. Power/ Die (W)	3.2	6.37	12.75

7.3.2 Matrix-Vector Applications

We evaluate the performance of the proposed 3DL-PIM architecture for three microtasks: Matrix-Vector Multiplication (GEMV), and Matrix-Matrix Multiplication (GEMM) with 8-bit precision for various input sizes (N=1-10000). These operations are mapped across four banks in a pCH in the MP-3DL-PIM configuration. Figure 6(b) presents the latencies of the GEMV and GEMM operations with different dimensions. It can be observed that the latency of the GEMV operation, which is memory-bound, scales up only linearly with increased dimension thanks to the elimination of stalling latencies from memory accesses.

Based on the implementation of large GEMM and GEMV operations, we also map Transformer in the proposed 3DL-PIM architecture. We evaluate performance for the BERT base architecture with twelve layers of transformer blocks and a hidden size of 768 and 12 self-attention heads. Our performance evaluations show that the HP, MP, and LP-3DL-PIM configurations respectively achieve 57 inputs/s, 114.5 inputs/s, and 229 inputs/s of throughput on BERT inferences with 8-bit fixed-point precision, with an energy-efficiency figure of 4.49 inputs/J. This makes it 22% more energy-efficient at BERT inferences compared to the Nvidia Tesla A100 GPU which achieves 4.39 inputs/J of computational energy efficiency at int8 operational precision [41]. The higher energy efficiency of 3DL-PIM compared to A100 can be attributed to the memory-bound nature of the BERT inference application which is characterized by very large GEMV and GEMM tasks with limited data re-use. Thanks to the proximity of data to the PEs within 3DL-PIM, it performs more efficient computing with minimal data re-use than the state-of-the-art GPU.

7.3.3 CNNs

We benchmark the parallel processing performance of all three configurations of 3DL-PIM for the inferences of the CNN algorithms: AlexNet, VGGNets (16, 19), GoogleNet, Inception V2, ResNets (18, 34, 50), and SqueezeNet for 8-bit and 4-bit fixed-point data-precisions. The input dimensions are 224×224×3 for all the CNNs, except for Inception V2 which has the input dimensions of 229×229×3. For all three of the 3DL-PIM hardware configurations, we map image threads to all the PIM-banks and

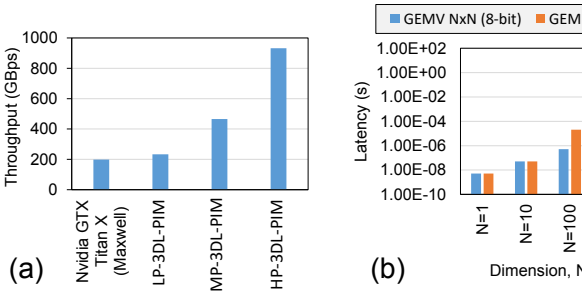


Fig. 6. Performance evaluation of 3DL-PIM for (a) 128-bit cryption and (b) Matrix and Vector arithmetic. The AES E performance is presented in terms of throughput (GBps) and p against Nvidia GTX Titan X GPU for reference. The performanc PIM for matrix/vector arithmetic applications is evaluated in latency by mapping each task across a single pCH of the MP configuration.

run identical instruction streams across all the BCUs in maximize the parallel throughput.

For ensuring maximum data reuse and optimization of performance, a specific data orchestration pattern can be adopted. We exclusively explore weight-stationary data orchestration in which the weight matrices of the DNNs are arranged at reserved locations in the subarrays and are never re-written whereas the distribution of the input/activation matrices/vectors in the subarray is flexible. As a result, while distributing large matrix-vector multiplications across multiple banks/chips in the system, only the input vector (or its portion) requires to be distributed across those chips. Thanks to the high bandwidth and low latency of the TSV-based communication, as shown in Table 3, this communication does not contribute a significant performance overhead.

Figure 7(a) presents the maximum achievable throughput of the three 3DL-PIM configurations for various CNN inferences with 8-bit and 4-bit precisions respectively. HP-3DL-PIM achieves a maximum throughput of respectively 1859.3 frames/s and 2953.5 frames/s for 8-bit and 4-bit fixed-point precision inferences of

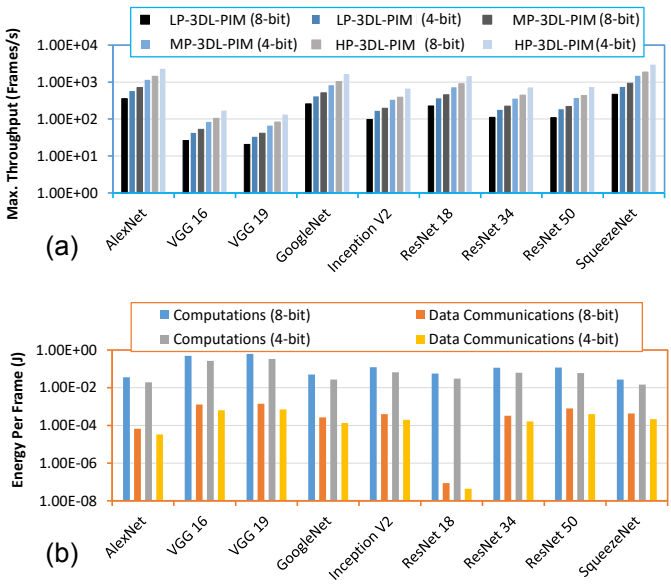


Fig. 7. Performance benchmark of the three configurations of 3DL-PIM, in terms of maximum (a) parallel throughput (frames/s) and (b) energy per frame (joule) with 8-bit and 4-bit fixed-point precision inferences for various CNN algorithms.

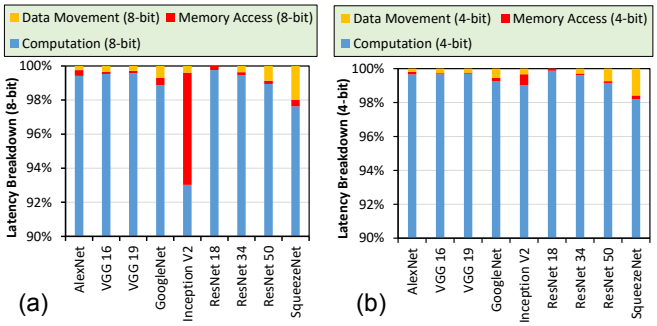


Fig. 8. Latency breakdown of a single inference of various CNN algorithms for 8-bit and 4-bit precisions, showing the relative latency of the computations inside the PEs, memory accesses by the PEs, and the data movement among the PEs in a Process Column for both cases.

largest CNN in comparison, are respectively 82.56 frames/s and 132.2 frames/s. The 4-bit fixed-point precision inferences offer nearly 60% higher parallel performance than the 8-bit precision inferences. Although performing CNN inferences with 4-bit fixed-point data precision inevitably affects the inference accuracies, it has been shown that the loss of accuracy can be minimized by using advanced quantization techniques [42]. Figure 7(b) presents the breakdown of energy consumption from the computational workload as well as the data communications during a single frame of inference with respectively 8-bit and 4-bit fixed-point precisions for various CNNs. It can be observed that the energy consumption from the data communications is at least two orders lower than the energy consumption from the computations across all CNNs. Such energy efficiency in the data communications is attained by restricting data movements only within the respective Process Columns during processing

Figure 8 demonstrates the optimization of data accesses and movement latencies by virtue of the proposed data-communication scheme within the Process Columns. Figures8 (a) and (b) show inference latency breakdown for a single frame of CNN inference with 8-bit and 4-bit data precision, respectively. During inferences, the output feature maps of the CNN layers are efficiently redistributed to different subarrays within the Process Column via the ultra-high-bandwidth subarray interlinks. As a result, the data-movement latency does not account for more than 2% of the overall inference latency for the benchmarked CNNs.

7.4 Comparative Evaluation

7.4.1 Comparison with Traditional Processing Devices

We compare 3DL-PIM with several high-end traditional computing devices (*i.e.* CPU and GPU) for ResNet-50 and VGG-19 inferences with 8-bit fixed-point precision. Figure 9(a) presents a comparison of 3DL-PIM with CPUs: Intel Core i9-10920X), Xeon Gold 5218T, Xeon Platinum 8270, Xeon Platinum 8380, and a GPU: Nvidia TITAN-X, in terms of performance per unit power consumption (Frames/s-W) [43], [44]. It can be observed from Figure 9(a) that HP-3DL-PIM offers significantly more power-efficient performance compared to all the devices in comparison. For example, it achieves 130% higher performance for unit power consumption at 8-bit fixed-point inferences of ResNet-50, compared to Intel Xeon Platinum 8380, a state-of-the-art high-performance server-grade CPU. HP-3DL-PIM also achieves higher energy efficiency for both ResNet-50 inferences (4.8×) and VGG-19 inferences (1.47×) than the GPU for a single stream of images. The GPU becomes more energy-efficient for a larger batch size of 8. This is because, for larger batch sizes (*i.e.* 8), the GPU is capable of pipelining multiple inference threads

in each shader core and therefore achieving improved energy efficiency at processing [44]. However, such sophistication also comes with a significantly larger chip footprint (942 mm^2) and power consumption (*i.e.* 250W) than the proposed memory-centric processing architecture with a power consumption of only 4-13W per stacked die (92 mm^2). However, the comparative energy efficiency is highly subjective to the nature of the application domain as well. For example, as discussed previously in Sections 7.3.1 and 7.3.2 respectively, the proposed 3DL-PIM is 22.9× more energy-efficient at AES encryption than the TITAN X GPU and 22% more efficient at BERT inferences with 8-bit precision than the Tesla A100 GPU.

7.4.2 Comparison with Memory-Based Accelerators

We also compare 3DL-PIM with several memory-centric accelerators, including 2-D DRAM-based in-situ AI accelerators DRISA [2] and SCOPE [3], 3-D stacked memory-based near-memory AI accelerators NeuroCube [23] and Tetris [24], and HBM2-based accelerator called FIMDRAM [7], [8] in terms of raw computing performance (TOPs/s) normalized to a unit area (mm^2) and unit thermal design power (TDP) in Watts in Figure 9(b). We use HP-3DL-PIM, the highest-performance variant of the proposed 3DL-PIM architecture for this comparison. The performance parameters for all the devices are normalized to the 22nm technology node.

It can be observed from Figure 9(b) that 3DL-PIM offers noticeably more energy-efficient and area-efficient computing performance compared to all the devices in comparison. Both DRISA and SCOPE are massively parallel bitwise processing architectures with very high processing throughput (1.06 TOPs/s and 7.08 TOPs/s, respectively). However, these architectures incur a high memory access rate for bitwise processing which makes these susceptible to high DRAM leakages. Also, there is a significant dynamic power dissipation from a large number of CMOS logic gates operating in parallel, resulting in lower area efficiency and energy-efficiency than 3DL-PIM. NeuroCube and Tetris place PEs only on the base logic die of the 3-D stacked HMC memory architecture. Alongside having relatively large PEs, these devices also incorporate on-chip Networks (NoC) to interconnect the PEs which, in turn, accounts for a significant area overhead and additional power dissipation. FIMDRAM [8], which is also the base design for the commercially designed Aquabolt-XL [7], [13], incorporates large floating point ALUs on the stacked memory chips in a near-bank layout, with 2.65× larger PEs than 3DL-PIM. But as discussed previously in Section 6, this architecture can only

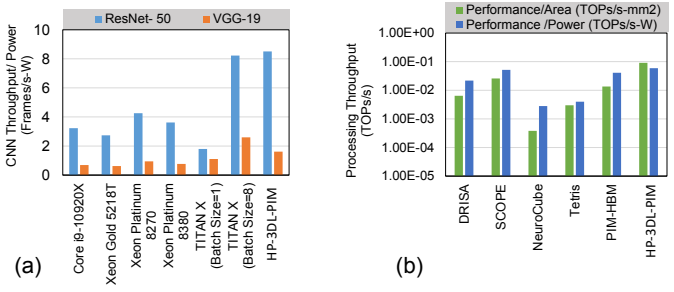


Fig. 9. Performance Comparison of 3DL-PIM with (a) several high-end CPUs and a GPU in terms of ResNet-50 and VGG-19 inference throughput normalized for unit power consumption, and (b) memory-centric accelerators DRISA, SCOPE, NeuroCube, Tetris, and Samsung's PIM-HBM in terms of performance per unit area (TOPs/s-mm^2) and unit power consumption (TOPs/s-W).

TABLE 5
Comparison of PE attributes of DRAM-based PIMs

Device	#PEs	Area (mm^2)	Power (W)	MAC Delay(ns)	Power× Delay(nJ)
DRISA -3T1C	32768	0.001	0.003	1768	5.3
DRISA -1T1C-NOR	16384	0.0025	0.006	2110	12.66
SCOPE Vanilla	65536	0.0028	0.0026	56	0.145
SCOPE H2D	65536	0.0029	0.0026	200	0.52
FIMDRAM	128	0.721	N/A	1.67	N/A
HP -3DL-PIM	**8192**	**0.021**	**0.0062**	**5.04**	**0.031**

chip, resulting in 2.6× lower overall processing throughput and 6.76× lower performance per unit power than HP-3DL-PIM.

Table 5 presents a quantitative comparison of the PEs of DRISA, SCOPE, FIMDRAM, and the proposed 3DL-PIM in. It can be observed that the bitwise processing PIMs, DRISA, and SCOPE are characterized by a large number of very small PEs that are essentially formed of a group of bitwise logic gates each. On the other hand, FIMDRAM, which is a near-bank accelerator, features large PEs located outside memory banks. Each PE contains 16 floating-point ALUs (FPUs) for 16-way parallel operation. Further, these FPUs can pipeline their operation to achieve the lowest operational latency of all the PIM devices in comparison. In contrast, the proposed 3DL-PIM has a significantly larger number of smaller (41.8×) PE footprints than FIMDRAM. Conveniently, the power consumption of a 3DL-PIM PE is also very low and comparable to the bitwise processors, DRISA and SCOPE. This, along with the low operational latency of 3DL-PIM allows it to achieve the lowest power-delay-product (PDP) of all the devices in comparison.

We also compare the proposed 3DL-PIM with another LUT-based memory-centric accelerator on 3-D stacked memory called DLUX [10]. DLUX is a near-bank accelerator that combines LUT-based computing with CMOS logic to support full-precision FP multiplications. Aimed primarily at Deep Learning Training applications, DLUX incorporates large LUT-based Process Engines (PE) in a 1:1 ratio with the number of banks. Although DLUX can support FP computations, a large portion of the FP computation is carried out with the aid of CMOS logic, alongside the LUT-based computations. In contrast, the proposed 3DL-PIM features a significantly larger number of compact PEs (*i.e.* 64 PEs/bank) that leverage only LUTs to perform fixed-point precision computing. These PEs are 19.7× smaller than the Process Engines of DLUX. A detailed comparison of these two architectures is presented in Table 6. Thanks to the within-the-bank placement of PEs, 3DL-PIM achieves remarkably lower data-access energy during computations (<8%) compared to DLUX (66.1%). Alongside, 3DL-PIM can expose a wider datapath and data bandwidth to its PEs by accessing data directly from the memory bitlines within the banks. Also, the collective energy consumption/ALU access for each bank is 13.8× lower for 3DL-PIM than DLUX.

Alongside DLUX, several recent memory-centric accelerators such as iPIM [17], and AxRAM [28] also leverage the near-bank computing layout. Compared to DLUX, iPIM, and AxRAM, the proposed 3DL-PIM has respectively 221×, 103×, and 4.73× lower energy consumption (pJ) per access of a parallel processing queue. The footprint of a processing queue as well as the overall

TABLE 6
Architectural Comparison of DLUX [10] and the Proposed 3DL-PIM

Attributes	DLUX	3DL-PIM
LUT Area (mm^2)	0.0095	0.002
ALU/Bank (mm^2)	2	64
PE Area (mm^2)	0.4146	0.021
ALU Area (mm^2)	(MVMx1/ VUx1/ PU x1) 0.244/0.1172/0.004	(LUT-core x1) 0.002
Area Overhead	34%	9.53/19/38.1 %
PE Memory (Kb)	32	0.5
Energy/Access (pJ)	(MVM/VU/PU) 139.96/3.6/64.3	(PE/Process Column) 0.06 /0.941

computing area overhead per chip of 3DL-PIM are also comparable with these architectures. For example, while DLUX and iPIM have respectively 34.02% and 10.71% area overheads, depending on the hardware configuration, the computing circuitry in 3DL-PIM occupies between 9.53% and 38.1% of chip area.

8 Conclusions

In this work, we present 3DL-PIM, a highly flexible, LUT-based Processing in Memory architecture implemented on the 3-D stacked memory platforms that can offer massively parallel processing performance on a wide range of memory-intensive applications. We develop a novel LUT-based programmable Processing Element architecture that can facilitate a wide range of logic/arithmetic operations at low power consumption. A large number of such Processing Elements are integrated within the memory banks to maximize bandwidth utilization and support massively parallel computing. We benchmark the hardware for data encryption, AI acceleration, and linear algebraic applications. Our hardware simulations demonstrate that the proposed 3DL-PIM architecture can offer up to 2.6× superior processing performance compared to state-of-the-art 3-D stacked memory-based near-bank accelerator, thanks to its 2.65× smaller and similarly energy-efficient LUT-based ALUs.

Acknowledgments

This work was supported in part by the US National Science Foundation (NSF) CAREER Grant CNS-1553264 and NSF Grant CNS-2228239.

References

[1] 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," ser. GLSVLSI '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 89–94. [Online]. Available: https://doi.org/10.1145/3386263.3407649

[2] S. Li, D. Niu, K. T. Malladi, H. Zheng, B. Brennan, and Y. Xie, "Drisa: A dram-based reconfigurable in-situ accelerator," in *2017 50th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, Oct 2017, pp. 288–301.

[3] S. Li, A. O. Glova, X. Hu, P. Gu, D. Niu, K. T. Malladi, H. Zheng, B. Brennan, and Y. Xie, "Scope: A stochastic computing engine for dram-based in-situ accelerator," in *2018 51st Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, Oct 2018, pp. 696–709.

[4] 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 *2019 56th ACM/IEEE Design Automation Conference (DAC)*, 2019, pp. 1–6.

[5] 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.

[6] T. P. Morgan, "Accelerating compute by cramming it into dram memory," Oct 2019. [Online]. Available: https://www.upmem.com/nextplatform-com-2019-10-03-accelerating-compute-by-cramming-it-into-dram/

[7] 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.

[8] S. Lee, S.-h. Kang, J. Lee, H. Kim, E. Lee, S. Seo, H. Yoon, S. Lee, K. Lim, H. Shin, J. Kim, O. Seongil, A. Iyer, D. Wang, K. Sohn, and N. S. Kim, "Hardware architecture and software stack for pim based on commercial dram technology : Industrial product," in *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*, 2021, pp. 43–56.

[9] M. He, C. Song, I. Kim, C. Jeong, S. Kim, I. Park, M. Thottethodi, and T. N. Vijaykumar, "Newton: A dram-maker's accelerator-in-memory (aim) architecture for machine learning," in *2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2020, pp. 372–385.

[10] 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*, pp. 1–1, 2020.

[11] O. Villa, D. R. Johnson, M. O'Connor, E. Bolotin, D. Nellans, J. Luitjens, N. Sakharnykh, P. Wang, P. Micikevicius, A. Scudiero, S. W. Keckler, and W. J. Dally, "Scaling the power wall: A path to exascale," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '14. IEEE Press, 2014, p. 830–841. [Online]. Available: https://doi.org/10.1109/SC.2014.73

[12] A. Nowatzyk, Fong Pong, and A. Saulsbury, "Missing the memory wall: The case for processor/memory integration," in *23rd Annual International Symposium on Computer Architecture (ISCA'96)*, May 1996, pp. 90–90.

[13] J. H. Kim, S.-h. Kang, S. Lee, H. Kim, W. Song, Y. Ro, S. Lee, D. Wang, H. Shin, B. Phuah, J. Choi, J. So, Y. Cho, J. Song, J. Choi, J. Cho, K. Sohn, Y. Sohn, K. Park, and N. S. Kim, "Aquabolt-xl: Samsung hbm2-pim with in-memory processing for ml accelerators and beyond," in *2021 IEEE Hot Chips 33 Symposium (HCS)*, 2021, pp. 1–26.

[14] "Hybrid memory cube – hmc gen2," https://www.micron.com/-/media/client/global/documents/products/data-sheet/hmc/gen2/hmc_gen2.pdf, (Accessed on 02/03/2022).

[15] D. U. Lee, K. W. Kim, K. W. Kim, H. Kim, J. Y. Kim, Y. J. Park, J. H. Kim, D. S. Kim, H. B. Park, J. W. Shin, J. H. Cho, K. H. Kwon, M. J. Kim, J. Lee, K. W. Park, B. Chung, and S. Hong, "25.2 a 1.2v 8gb 8-channel 128gb/s high-bandwidth memory (hbm) stacked dram with effective microbump i/o test methods using 29nm process and tsv," in *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, 2014.

[16] J. Jeddeloh and B. Keeth, "Hybrid memory cube new dram architecture increases density and performance," in *2012 Symposium on VLSI Technology (VLSIT)*, 2012, pp. 87–88.

[17] P. Gu, X. Xie, Y. Ding, G. Chen, W. Zhang, D. Niu, and Y. Xie, "ipim: Programmable in-memory image processing accelerator using near-bank architecture," in *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*, 2020, pp. 804–817.

[18] C. Sudarshan, M. H. Sadi, C. Weis, and N. Wehn, "Optimization of dram based pim architecture for energy-efficient deep neural network training," in *2022 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2022, pp. 1472–1476.

[19] J. Wang, X. Wang, C. Eckert, A. Subramaniyan, R. Das, D. Blaauw, and D. Sylvester, "14.2 a compute sram with bit-serial integer/floating-point operations for programmable in-memory vector acceleration," in *2019 IEEE International Solid- State Circuits Conference - (ISSCC)*, 2019, pp. 224–226.

[20] 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.

[21] 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 Transactions on Parallel and Distributed Systems*, pp. 1–1, 2021.

[22] P. R. Sutradhar, K. Basu, S. M. P. Dinakarrao, and A. Ganguly, "An ultra-efficient look-up table based programmable processing in memory

architecture for data encryption," in *2021 IEEE 39th International Conference on Computer Design (ICCD)*, 2021, pp. 252–259.

[23] D. Kim, J. Kung, S. Chai, S. Yalamanchili, and S. Mukhopadhyay, "Neurocube: A programmable digital neuromorphic architecture with high-density 3d memory," in *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*, 2016, pp. 380–392.

[24] M. Gao, J. Pu, X. Yang, M. Horowitz, and C. Kozyrakis, "Tetris: Scalable and efficient neural network acceleration with 3d memory," in *Proceedings of the Twenty-Second International Conference on Architectural Support for Programming Languages and Operating Systems*, ser. ASPLOS '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 751–764. [Online]. Available: https://doi.org/10.1145/3037697.3037702

[25] B. K. Joardar, A. I. Arka, J. R. Doppa, P. P. Pande, H. Li, and K. Chakrabarty, "Heterogeneous manycore architectures enabled by processing-in-memory for deep learning: From cnns to gnns: (iccad special session paper)," in *2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, 2021, pp. 1–7.

[26] 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 *2016 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, March 2016, pp. 568–580.

[27] D. Jeon, K. Park, and K. Chung, "Hmc-mac: Processing-in memory architecture for multiply-accumulate operations with hybrid memory cube," *IEEE Computer Architecture Letters*, vol. 17, no. 1, pp. 5–8, Jan 2018.

[28] A. Yazdanbakhsh, C. Song, J. Sacks, P. Lotfi-Kamran, H. Esmaeilzadeh, and N. S. Kim, "In-dram near-data approximate acceleration for gpus," in *Proceedings of the 27th International Conference on Parallel Architectures and Compilation Techniques*, ser. PACT '18. New York, NY, USA: Association for Computing Machinery, 2018. [Online]. Available: https://doi.org/10.1145/3243176.3243188

[29] X. Xie, Z. Liang, P. Gu, A. Basak, L. Deng, L. Liang, X. Hu, and Y. Xie, "Spacea: Sparse matrix vector multiplication on processing-in-memory accelerator," in *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, 2021, pp. 570–583.

[30] S. Kim, S. Kim, K. Cho, T. Shin, H. Park, D. Lho, S. Park, K. Son, G. Park, and J. Kim, "Processing-in-memory in high bandwidth memory (pim-hbm) architecture with energy-efficient and low latency channels for high bandwidth system," in *2019 IEEE 28th Conference on Electrical Performance of Electronic Packaging and Systems (EPEPS)*, 2019, pp. 1–3.

[31] Y. Kim, V. Seshadri, D. Lee, J. Liu, and O. Mutlu, "A case for exploiting subarray-level parallelism (salp) in dram," in *2012 39th Annual International Symposium on Computer Architecture (ISCA)*, June 2012, pp. 368–379.

[32] J. H. Cho, J. Kim, W. Y. Lee, D. U. Lee, T. K. Kim, H. B. Park, C. Jeong, M.-J. Park, S. G. Baek, S. Choi, B. K. Yoon, Y. J. Choi, K. Y. Lee, D. Shim, J. Oh, J. Kim, and S.-H. Lee, "A 1.2v 64gb 341gb/s hbm2 stacked dram with spiral point-to-point tsv structure and improved bank group data control," in *2018 IEEE International Solid - State Circuits Conference - (ISSCC)*, 2018, pp. 208–210.

[33] B. K. Joardar, A. I. Arka, J. R. Doppa, and P. P. Pande, "3d++: Unlocking the next generation of high-performance and energy-efficient architectures using m3d integration," in *2021 Design, Automation Test in Europe Conference Exhibition (DATE)*, 2021, pp. 158–163.

[34] F. Gao, G. Tziantzioulis, and D. Wentzlaff, "Computedram: In-memory compute using off-the-shelf drams," ser. MICRO '52. New York, NY, USA: Association for Computing Machinery, 2019, p. 100–113. [Online]. Available: https://doi.org/10.1145/3352460.3358260

[35] V. Seshadri, K. Hsieh, A. Boroum, D. Lee, M. A. Kozuch, O. Mutlu, P. B. Gibbons, and T. C. Mowry, "Fast bulk bitwise and and or in dram," *IEEE Computer Architecture Letters*, vol. 14, no. 2, pp. 127–131, July 2015.

[36] N. Chatterjee, M. O'Connor, D. Lee, D. R. Johnson, S. W. Keckler, M. Rhu, and W. J. Dally, "Architecting an energy-efficient dram system for gpus," in *2017 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, 2017, pp. 73–84.

[37] S. S. N. Larimi, B. Salami, O. S. Unsal, A. C. Kestelman, H. Sarbazi-Azad, and O. Mutlu, "Understanding power consumption and reliability of high-bandwidth memory with voltage underscaling," *CoRR*, vol. abs/2101.00969, 2021. [Online]. Available: https://arxiv.org/abs/2101.00969

[38] 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

[39] Y. Eckert, N. Jayasena, and G. H. Loh, "Thermal feasibility of die-stacked processing in memory," 2014. [Online]. Available: https://www.semanticscholar.org/paper/Thermal-Feasibility-of-Die-Stacked-Processing-in-Eckert-Jayasena/a52945840b980adfef34466cb4186c7cda3b61e6

[40] 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

[41] "Nvidia data center deep learning product performance," Mar 2023. [Online]. Available: https://developer.nvidia.com/deep-learning-performance-training-inference

[42] Y. Choukroun, E. Kravchik, F. Yang, and P. Kisilev, "Low-bit quantization of neural networks for efficient inference." in *ICCV Workshops*, 2019, pp. 3009–3018.

[43] [Online]. Available: https://docs.openvino.ai/2023.0/openvino_docs_performance_benchmarks.html

[44] "8-bit inference with tensorrt." april 2021. [Online]. Available: https://on-demand.gputechconf.com/gtc/2017/presentation/s7310-8-bit-inference-with-tensorrt.pdf

BIOGRAPHIES

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.

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 an 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. She is an active student IEEE member.

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.

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.