MPC-Wrapper: Fully Harnessing the Potential of Samsung Aquabolt-XL HBM2-PIM on FPGAs

Jinwoo Choi*[1], Yeonan Ha*[1], Hanna Cha*, Seil Lee*, Sungchul Lee*, Jounghoo Lee*,
Shin-haeng Kang†, Bongjun Kim‡, Hanwoong Jung‡, Hanjun Kim*, and Youngsok Kim*§
*Yonsei University †Samsung Electronics ‡Samsung Advanced Institute of Technology
{jinwoo1029, yeonan, hanna.cha, seil.lee, sungchul_lee, jounghoolee}@yonsei.ac.kr,
{s-h.kang, bj90.kim, hw7884.jung}@samsung.com, {hanjun, youngsok}@yonsei.ac.kr

***Abstract*—Processing-In-Memory (PIM) is an attractive solution for mitigating frequent and large data movement between computational units and memory devices. Among various PIM implementations, Samsung Aquabolt-XL is an HBM2 memory device which implements 16 PIM-enabled pseudo-channels and associates an In-Memory Processor (IMP) to each pair of the memory banks. Recent studies have shown that Aquabolt-XL can greatly accelerate various applications (e.g., deep learning) by offloading memory-intensive operations (e.g., matrix-vector multiplications) to the IMPs. However, the prior study fails to fully utilize Aquabolt-XL and achieves limited performance gains by offloading operations to the IMPs of only a single pseudo-channel. Ideally, utilizing all the 16 pseudo-channels of Aquabolt-XL can further accelerate the key operations by a factor of 16× compared to utilizing only a single pseudo-channel. To fully exploit Aquabolt-XL, therefore, memory-intensive operations should be offloaded to and concurrently executed on the IMPs of all the PIM-enabled pseudo-channels.**

This paper presents *MPC-Wrapper*, a multi-pseudo-channel wrapper interface which allows memory-intensive operations to be offloaded to and concurrently executed on the IMPs of all the 16 PIM-enabled pseudo-channels of Aquabolt-XL. First, MPC-Wrapper allows all the PIM-enabled pseudo-channels to operate independently and in parallel, thus achieving high scalability needed for fully utilizing all the PIM-enabled pseudo-channels of Aquabolt-XL. Second, MPC-Wrapper is highly flexible as it exposes the PIM-enabled pseudo-channels as separate ports and enables an FPGA logic to flexibly utilize any set of the PIM-enabled pseudo-channels according to its needs. Third, MPC-Wrapper achieves high usability by hiding the complex low-level interactions between the memory controller and Aquabolt-XL for initializing and invoking the PIM-enabled pseudo-channels from the other FPGA logics. Using an Aquabolt-XL-equipped Xilinx Alveo U280 FPGA and four memory-intensive benchmarks, we show that utilizing all the 16 PIM-enabled pseudo-channels of Aquabolt-XL with MPC-Wrapper achieves a geometric mean speedup of 13.66× over the baseline single PIM-enabled pseudo-channel implementations of the benchmarks.

I. Introduction

Emerging memory-intensive applications, large-scale deep learning (e.g., recommendation [1], translation [2]) in particular, incur frequent memory accesses and extensively utilize large-scale datasets. Among possible solutions for mitigating the memory wall, Processing-In-Memory (PIM) has emerged as a practical solution as it can significantly reduce the data movement between memory devices and computational units [3]–[13]. PIM-enabled memory devices [14]–[17] implement In-Memory Processors (IMPs) located near their memory banks and allow memory-intensive operations to be offloaded onto the IMPs. The offloaded operations then can directly access the data stored in the memory banks, allowing applications to exploit the abundant bank-level parallelism of the memory devices. For example, recent studies [16]–[18] employ Aquabolt-XL [16], [17], Samsung's commodity High-Bandwidth Memory (HBM)-based PIM-enabled memory device, on Field-Programmable Gate Arrays (FPGAs) and accelerate memory-intensive deep learning applications (e.g., RNN-T [19]) by offloading element-wise additions and matrix-vector multiplications from the FPGA logic onto the IMPs.

Despite achieving large performance improvements, we find that the existing PIM acceleration study [18] using Aquabolt-XL fails to fully exploit the memory device and thus achieves limited performance, scalability, and applicability. The existing study offloads their target memory-intensive operations onto the IMPs of only a single PIM-enabled pseudo-channel of Aquabolt-XL. However, Aquabolt-XL is equipped with a total of 16 PIM-enabled pseudo-channels and 128 IMPs, and the existing study limits its performance gains and scalability. Ideally, utilizing all the 16 PIM-enabled pseudo-channels of Aquabolt-XL would provide an additional speedup of 16× over utilizing only a single PIM-enabled pseudo-channel. In addition, the existing study implements and handles all the complex low-level interactions with Aquabolt-XL's pseudo-channel within the application-side FPGA logic. Imposing the huge burden of considering and implementing all the complex low-level interactions to applications significantly degrades the applicability and generality of the existing study.

In this paper, we present *MPC-Wrapper*, a multi-pseudo-channel wrapper interface designed to maximize the benefits of Aquabolt-XL on FPGAs. To the best of our knowledge, this paper is the first study fully harnessing all the 16 PIM-enabled pseudo-channels of Aquabolt-XL with the FPGA. MPC-Wrapper allows memory-intensive applications to fully utilize all the available PIM-enabled pseudo-channels of Aquabolt-XL as follows. First, MPC-Wrapper allows the PIM-enabled pseudo-channels to operate independently and in parallel. By utilizing all the PIM-enabled pseudo-channels, MPC-Wrapper overcomes the prior study's limited performance improvements and scalability achieved with only a single PIM-enabled

[1]Co-first authors
[§]Corresponding author

2576-2621/24/$31.00 ©2024 IEEE
DOI 10.1109/FCCM60383.2024.00027

pseudo-channel. Second, MPC-Wrapper exposes all the PIM-enabled pseudo-channels as separate ports to the FPGA logic. According to its needs, the FPGA logic can flexibly utilize any set of the PIM-enabled pseudo-channels, and thus can execute varying PIM operations on different sets of the PIM-enabled pseudo-channels. Third, MPC-Wrapper hides all the complex low-level interactions between the memory controller and Aquabolt-XL from the FPGA logic. This allows MPC-Wrapper to achieve high usability as the applications can easily offload their operations to the PIM-enabled pseudo-channels without implementing the low-level interactions.

Using an Aquabolt-XL-equipped Xilinx Alveo U280 FPGA, we show that MPC-Wrapper makes memory-intensive applications easily exploit the large potential of Aquabolt-XL. We demonstrate the high effectiveness of MPC-Wrapper using three key operations of deep learning models (embedding lookup, matrix-vector multiplication, matrix-matrix multiplication) and an end-to-end deep learning model (Alibaba's recommendation model [20]). To provide a useful guideline on accelerating memory-intensive applications with Aquabolt-XL, we present in-depth descriptions on how the four benchmarks have been implemented and parallelized to utilize multiple PIM-enabled pseudo-channels. In particular, we show that overlapping the executions of the FPGA logic and the PIM-enabled pseudo-channels is essential for mitigating the synchronous operating architecture of the PIM-enabled pseudo-channels.

Our evaluation using the Aquabolt-XL-equipped FPGA and the four benchmarks first shows that MPC-Wrapper achieves a geometric mean speedup of 13.66× over the baseline PIM implementations, which utilize only a single PIM-enabled pseudo-channel, by exploiting all the 16 PIM-enabled pseudo-channels. The evaluation then shows that, compared to the non-PIM implementations utilizing 16 pseudo-channels of the standard HBM2 memory device, MPC-Wrapper achieves a geometric mean speedup of 5.19×. After that, the evaluation demonstrates MPC-Wrapper's ability to concurrently accelerate multiple benchmarks without incurring any performance degradation due to resource sharing by allocating disjoint sets of the PIM-enabled pseudo-channels to the benchmarks.

In summary, this paper makes the following contributions:

- We show that the prior study on Aquabolt-XL utilizes only a single PIM-enabled pseudo-channel, and thus attains limited performance, scalability, and applicability.
- We propose MPC-Wrapper, a flexible and scalable multi-pseudo-channel wrapper interface for Aquabolt-XL which allows all the 16 pseudo-channels of Aquabolt-XL to operate independently and in parallel.
- We reveal that, with MPC-Wrapper, overlapping the executions of the FPGA logic and the PIM-enabled pseudo-channels is essential to mitigate the synchronous operating architecture of the PIM-enabled pseudo-channels.
- We implement MPC-Wrapper on an Aquabolt-XL-equipped Xilinx Alveo U280 FPGA and show that MPC-Wrapper achieves significant performance improvements over the baseline single PIM-enabled pseudo-channel implementations of the four representative benchmarks.

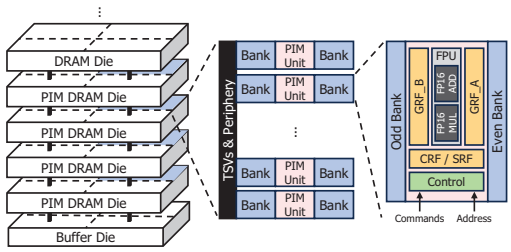


Fig. 1: The internal architecture of Samsung Aquabolt-XL

II. Background

A. Processing-In-Memory & Samsung Aquabolt-XL

Processing-In-Memory (PIM) is a promising solution for overcoming the memory wall incurred by frequent data movement between memory devices and computational units [3]–[13]. Among a few commercial memory devices supporting PIM, Samsung Aquabolt-XL [16], a PIM-enabled HBM2 memory device, places PIM units near its memory banks to support PIM. Fig. 1 depicts the internal architecture of Aquabolt-XL. It comprises a buffer die, four PIM-DRAM dies, and four standard DRAM dies. The buffer die interacts with the PIM-DRAM and the standard DRAM dies through Through-Silicon Vias (TSVs). The PIM-DRAM dies and the standard DRAM dies contain 64 memory banks each. Within each of the four PIM-DRAM dies, a pair of the memory banks is associated with a PIM unit. The PIM unit consists of three major components: a 16-bit Floating-Point Unit (FPU) capable of performing 16-wide Single-Instruction Multiple-Data (SIMD) additions and multiplications, three Register Files (RFs), and a control unit. The General RF (GRF) and the Scalar RF (SRF) store the input and output data of a SIMD operation. The Command RF (CRF) stores the instructions needed by the PIM unit to perform a series of SIMD operations. The control unit is in charge of retrieving the instructions and metadata from a memory controller, executing the instructions on using the FPU, and transferring the input and output data between the RFs and the memory banks.

Aquabolt-XL groups the memory banks of its PIM-DRAM and standard DRAM dies into 16 pseudo-channels. Each pseudo-channel comprises 16 memory banks of the PIM-DRAM dies and 16 memory banks of the standard DRAM dies. This makes Aquabolt-XL provide 16 PIM-enabled pseudo-channels, each consisting of eight PIM units. The pseudo-channels can operate independently and in parallel.

Aquabolt-XL can act as both a standard HBM2 memory device and a PIM device by configuring its pseudo-channels to operate as one of the three following modes. First, the Single-Bank (SB) mode makes a pseudo-channel act as that of a standard HBM2 memory device. The SB-mode pseudo-channels retrieve standard HBM2 commands (e.g., memory reads and writes) from their memory controller(s) and process the commands. Second, the All-Bank (AB) and the All-Bank-PIM (AB-PIM) modes are used for offloading operations onto the PIM units of a pseudo-channel. The memory controller first sets a pseudo-channel to the AB mode to configure the

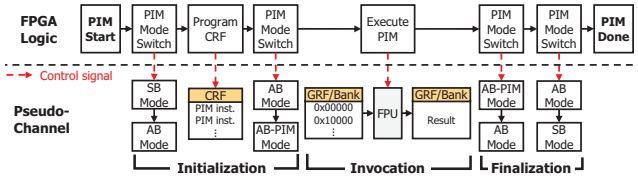
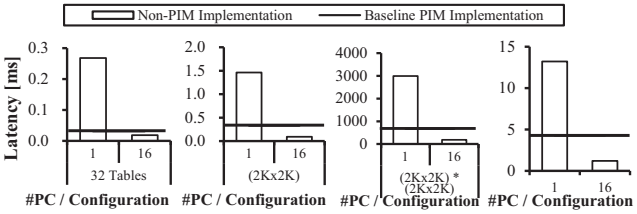


Fig. 2: Procedure for offloading computation from the FPGA logic onto the PIM units of Aquabolt-XL's pseudo-channel



(a) Embedding (b) GEMV (c) GEMM (d) AliRM

Fig. 3: Execution latency of benchmarks on the non-PIM implementation with various numbers of pseudo-channels (PC) and the baseline PIM implementation

PIM units (e.g., upload a series of instructions for the PIM units to their CRFs) and then switches the pseudo-channel to the AB-PIM mode to make the PIM units execute the uploaded instructions. In addition, due to the synchronous operating architecture of the pseudo-channels in the AB mode and the AB-PIM mode, all the memory banks of a pseudo-channel operate synchronously and perform the same instruction given to the pseudo-channel. Aquabolt-XL does not allow switching from the SB mode directly to the AB-PIM mode for a pseudo-channel; a pseudo-channel, currently in the SB mode, must be switched to the AB mode first and then to the AB-PIM mode.

B. Offloading Computation onto Aquabolt-XL's PIM Units

The FPGA logic can offload memory-intensive operations onto the PIM units by leveraging the three operating modes of Aquabolt-XL's pseudo-channels. Fig. 2 illustrates a typical procedure for offloading the operations onto the PIM units of a pseudo-channel. The procedure largely consists of three stages: initialization, invocation, and finalization. First, the initialization stage switches the pseudo-channel from the SB mode to the AB mode, uploads a series of PIM instructions to the CRFs, and then switches the pseudo-channel to the AB-PIM mode. Then, in the invocation stage, the FPGA logic sends a PIM invocation command to the PIM units which then execute the instructions stored in their CRFs. The instructions make the PIM units either perform FP16 additions/multiplications on the FPUs or transfer data between the GRFs and the memory banks. When the PIM units complete executing the instructions, the pseudo-channel notifies the FPGA logic of the completion. After that, the FPGA logic performs the finalization stage by switching the pseudo-channel to the SB mode and then reading out the output data stored in the memory banks. In the remainder of this paper, we call an FPGA IP block performing the described procedure as a PIM interface, since the procedure can be seen as an interface between the FPGA logic and the PIM units.

III. Motivation

Recently, Kang et al. [18] study using Aquabolt-XL with an FPGA. They accelerate RNN-T [19] by offloading the matrix-vector multiplication operations of RNN-T from the FPGA logic onto the PIM units of Aquabolt-XL. They show that offloading the operations to Aquabolt-XL can achieve large performance improvements over the baseline non-PIM implementations attaching a standard HBM2 memory device. Unfortunately, we find that the prior study [18], which utilizes Aquabolt-XL on the FPGA, achieves suboptimal performance, scalability, and applicability. The prior study focuses on only one (out of 16) pseudo-channel, and thus achieves limited performance and scalability by not fully exploiting the capabilities of Aquabolt-XL. To overcome the limitations, the FPGA logic should be able to utilize all the 16 pseudo-channels of Aquabolt-XL, enhancing performance and scalability.

Fig. 3 illustrates the limited performance encountered when employing only a single pseudo-channel of Aquabolt-XL. We measure the execution latency of key operations in deep learning models (embedding lookup, matrix-vector multiplication, matrix-matrix multiplication) and Alibaba's recommendation model (AliRM) on the baseline PIM implementation and the non-PIM implementation. The baseline PIM implementation uses a single pseudo-channel of Aquabolt-XL similar to the prior study [18], whereas the non-PIM implementation utilizes multiple pseudo-channels of the standard HBM2 memory device. The baseline PIM implementation shows faster execution latency than the non-PIM implementation employing a single pseudo-channel of the standard HBM2 memory device. However, it becomes slower when multiple pseudo-channels of the standard HBM2 memory device are utilized. This slowdown is attributed to the limited capabilities of a single pseudo-channel in Aquabolt-XL. Utilizing all the pseudo-channels of Aquabolt-XL ideally brings a speedup of 16× over utilizing only a single pseudo-channel. By using all pseudo-channels to overcome limited performance, Aquabolt-XL can enhance performance in memory-intensive operations.

To fully harness the potential of Aquabolt-XL, the FPGA logic has to execute multiple pseudo-channels independently and in parallel. Since Aquabolt-XL has scalability where the computational capability increases with the number of PIM units used, the host should be capable of executing operations across multiple pseudo-channels. However, the prior study's approach can not be naïvely extended to multiple pseudo-channels. When the FPGA logic uses multiple pseudo-channels, it must undergo an address partitioning process, separating instructions and data based on the number of pseudo-channels. This process involves converting them into pseudo-channel-specific addresses and distributing them accordingly. Consequently, the FPGA logic faces challenges in executing pseudo-channels in parallel, as the address partitioning process occurs each time the FPGA logic offloads operations to the pseudo-channel. To fully exploit the capability of Aquabolt-

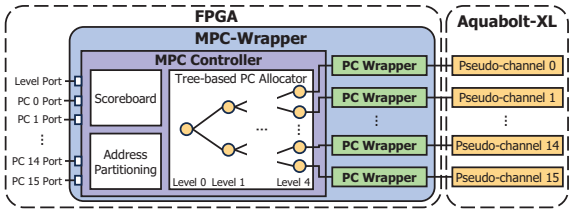


Fig. 4: Overview of MPC-Wrapper

XL, there is a need to offload the address partitioning process from the FPGA logic and execute it only once, rather than repeatedly, alleviating the burden of the FPGA logic and enabling efficient parallelization of pseudo-channels.

To fully exploit all the pseudo-channels of Aquabolt-XL, the FPGA logic must handle the PIM interface, which involves all the complex low-level interactions with Aquabolt-XL. As the usability of Aquabolt-XL bottoms out due to the PIM interface's overhead, the FPGA logic should be able to easily use each pseudo-channel through high-level abstraction. However, the prior study not only imposes the burden of implementing and handling the PIM interface on the FPGA logic but also lacks consideration for the adoption of multiple pseudo-channels. To address the limitation, there is a need to offload the PIM interface from the FPGA logic and introduce dedicated logic to manage the essential process, enhancing usability without considering the complex interactions.

IV. MPC-Wrapper

A. Design Goals & Overview

We propose Multiple-Pseudo-Channel Wrapper (MPC-Wrapper), a wrapper interface to fully harness the potential of Aquabolt-XL by offloading operations onto all the pseudo-channels and executing them independently and in parallel. To the best of our knowledge, MPC-Wrapper is the first study to utilize all the pseudo-channels of Aquabolt-XL on an FPGA. Motivated by the limitations of the prior study, we set three design goals for MPC-Wrapper. First, MPC-Wrapper must achieve scalability that increases the computational capabilities following the growth of the number of pseudo-channels. Second, MPC-Wrapper should allow the FPGA logic to flexibly utilize pseudo-channels according to its needs. Third, MPC-Wrapper has to reduce the load on the FPGA logic caused by managing the PIM interface.

Following these design goals, we construct MPC-Wrapper with MPC Controller and Pseudo-Channel Wrappers (PC Wrappers). Fig. 4 shows an overview of MPC-Wrapper. The MPC Controller, which manages PC Wrappers, provides scalability by enabling multiple pseudo-channels to operate independently and in parallel. It also offers flexibility by exposing all the pseudo-channels as separate ports, thus allowing the FPGA logic to dynamically use the desired number of them by configuring a set. The PC Wrappers enhance usability by offloading the PIM interface from the FPGA logic. MPC-Wrapper offers a seamless implementation with any FPGA logic, allowing the FPGA logic to freely offload operations to Aquabolt-XL through MPC-Wrapper. The generality provided by MPC-Wrapper allows for effective utilization of Aquabolt-XL regardless of a domain of the FPGA logic's target applications and/or functions, enhancing the overall performance.

B. MPC Controller

MPC Controller is positioned between the FPGA logic and the PC Wrappers, as illustrated in Fig. 4. To provide flexibility to the FPGA logic, the MPC Controller exposes all the pseudo-channels of Aquabolt-XL as separate ports. We expose as many `PC` ports as the number of pseudo-channels. The number of pseudo-channels required by the FPGA logic can vary depending on the operation being performed. Through the exposed ports, the MPC Controller enables the FPGA logic to flexibly utilize any set of the pseudo-channels (PC set) based on the desired number of pseudo-channels.

To configure a PC set, the MPC Controller employs a scoreboard and a tree-based pseudo-channel allocator (PC allocator). First, the scoreboard, employing a bit vector format, indicates which pseudo-channels are available to the FPGA logic through an exposed port. The MPC Controller updates the scoreboard when the status of each pseudo-channel is updated (i.e., busy or idle). Then, the FPGA logic can select available pseudo-channels by assessing the scoreboard of the MPC Controller and group them into a PC set. To easily allocate appropriate pseudo-channels and configure the PC set, we implement the PC allocator. As illustrated in Fig. 4, we design the PC allocator using a binary tree logic, where each pseudo-channel serves as a leaf node. We use a simple topology (i.e., binary tree) to reduce wiring overhead and achieve acceptable flexibility; implementing more fine-grained pseudo-channel allocation requires complex all-to-all topologies and incurs substantial wiring overhead. By providing a level value through `Level` port to the MPC Controller, the FPGA logic can easily configure the PC set its needs using the tree logic. The MPC Controller allocates pseudo-channels corresponding to leaf nodes of a subtree into the PC set. The subtree's root node is determined by selecting the node with the lowest index among the available nodes at the given level. For example, when the FPGA logic assigns 1 to the `Level` port, the PC set consists of pseudo-channels 0 to 7, corresponding to the leaf nodes of the subtree rooted at the level 1 node.

Through the `PC` ports, we can deploy instructions and data to the designated pseudo-channels. When the FPGA logic constructs a PC set, it only needs to deliver instructions and data through the `PC` port with the lowest index in the set, then the MPC Controller distributes them to the PC set. This eliminates the need to configure and deploy them to each individual PC port. Currently, with a single Aquabolt-XL, we expose 16 `PC` ports. When the FPGA logic hopes to use multiple Aquabolt-XLs, the MPC Controller exhibits its capability to seamlessly scale out the PC allocator, the scoreboard, and the number of `PC` ports to align with the total number of pseudo-channels.

With such flexibility, the FPGA logic can execute various PIM operations by allocating disjoint PC sets to different

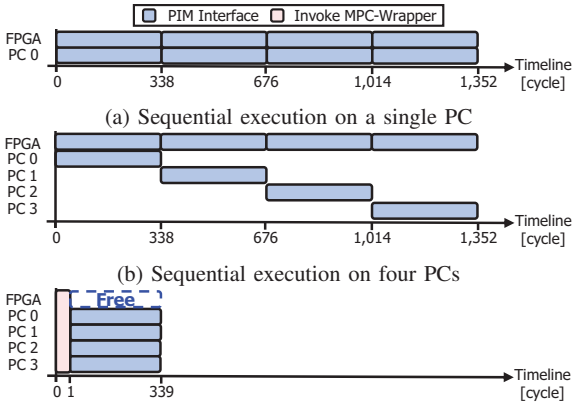


Fig. 5: Execution timelines of embedding lookup operation on four pseudo-channels (PCs)

operations. The FPGA logic only needs to specify the number of pseudo-channels it needs. It invokes MPC-Wrapper once with the deployment of instructions and data to the `PC` port having the lowest index of each PC set.

In the prior study, the FPGA logic needs to perform the address partitioning process each time to invoke the pseudo-channel, as shown in Fig. 5a. However, in case the FPGA logics incorrectly set the destination pseudo-channels of their PIM instructions, the instructions will be incorrectly delivered to the pseudo-channels, causing a memory address conflict which can break the functional correctness. The interconnect in Xilinx HBM IP [21] identifies the target pseudo-channel of memory requests using the five most-significant bits (MSBs) of its memory address. To avoid memory address conflicts, we implement the address partitioning process, which sets the five MSBs of the PIM instructions to the corresponding pseudo-channel's base address. Nevertheless, performance degradation does not occur because the prior study uses only a single pseudo-channel. However, when we naïvely scale out it to multiple pseudo-channels, the FPGA logic sequentially utilizes the pseudo-channels due to the address partitioning process at each time. Fig. 5b depicts the timeline, which fails to leverage the advantages of Aquabolt-XL's scalability. To harness Aquabolt-XL's scalability, we should execute multiple pseudo-channels independently and in parallel, as shown in Fig. 5c. By leveraging all the pseudo-channels, we can ideally achieve a speedup of 16× compared to utilizing a single pseudo-channel.

For the parallel execution, we employ an address partitioning logic within the MPC Controller. When multiple FPGA logics share Aquabolt-XL, memory address conflicts can occur. We design the address partitioning logic to prohibit memory address conflicts among pseudo-channels and automatically distribute instructions and data to the pseudo-channels. Following the characteristics of Xilinx HBM IP [21], the address partitioning logic of MPC-Wrapper remaps the MSBs of a memory request and redirects the PIM instructions to the appropriate pseudo-channels. By abstracting the address

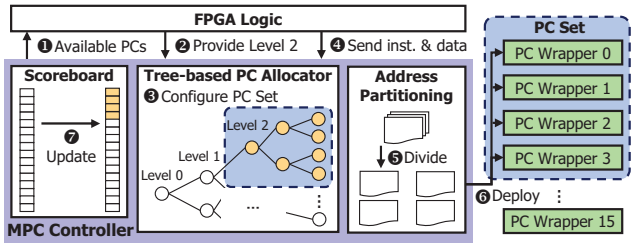


Fig. 6: Working model of MPC Controller

partitioning process, the FPGA logic doesn't need to perform the address partitioning process at each time. The MPC Controller enables the execution of pseudo-channels simultaneously by the address partitioning logic and broadcasting the start signal to them. When the FPGA logic executes the PC set and deploys instructions and data, the MPC Controller divides them into addresses corresponding to each pseudo-channel within the PC set. Therefore, MPC-Wrapper can operate all the pseudo-channels through the parallel execution.

In the example scenario shown in Fig. 5c, the parallel execution with four pseudo-channels by the MPC Controller significantly reduces the execution latency of the embedding lookup operation, resulting in almost 4× speedup, from 1,352 cycles down to 339 cycles. To execute pseudo-channels in parallel, the procedure of the scoreboard, the PC allocator, and the address partitioning is essential. However, it generates a negligible overhead, just one cycle. The scoreboard takes a cycle to update its bit vector and requires a register to hold the status of pseudo-channels which depends on prior PIM requests. The PC allocator is implemented using MUXes to enable routing. Address partitioning logic is implemented by hard-wiring an FPGA logic's AXI requests to the assigned pseudo-channels. So, the PC allocator and address partitioning logic can be implemented as combinational logic. Therefore, the parallel execution ensures scalability with effective utilization of pseudo-channels, enhancing overall performance.

Fig. 6 illustrates a working model of the MPC Controller in the same scenario as Fig. 5c. ❶ The FPGA logic first assesses the scoreboard to check which pseudo-channels are available. After that, ❷ the FPGA logic provides 2 to the `Level` port, and ❸ the PC allocator constructs a PC set with pseudo-channels from 0 to 3. ❹ The FPGA logic sends instructions and data into `PC 0` port, and ❺ the address partitioning logic divides them according to the allocated address of each pseudo-channel. ❻ The address partitioning logic deploys divided instructions and data across the pseudo-channel 0 to 3 of the PC set. ❼ After the MPC Controller updates the scoreboard, it invokes the PC Wrappers.

C. PC Wrappers

The PC Wrappers of MPC-Wrapper provide high usability to the FPGA logic by performing the PIM interface instead of the FPGA logic. The PIM interface, described in Section II-B as the complex low-level interactions, is an essential process to use pseudo-channels. In the prior study, the FPGA logic has to continuously handle the PIM interface.

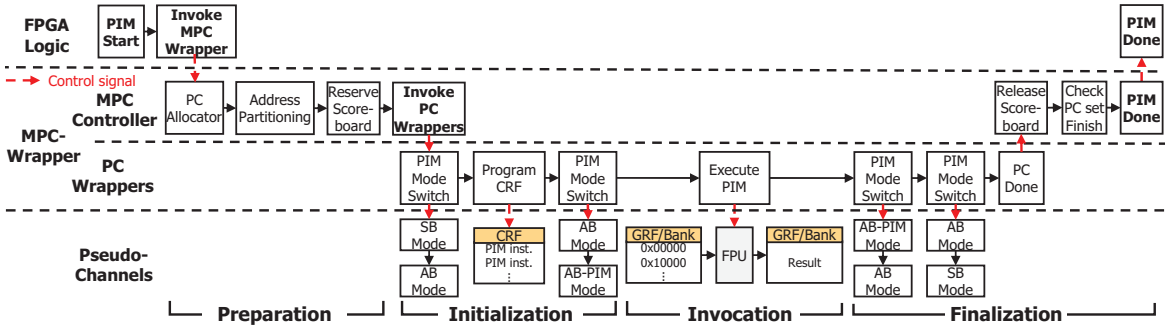


Fig. 7: Control flow of MPC-Wrapper

To provide high usability, we design the PC Wrappers by implementing the procedure of the PIM interface as a dedicated Finite State Machine (FSM) logic. The FSM logic includes the same three stages of the PIM interface performed by the FPGA logic: initialization, invocation, and finalization stage. When the MPC Controller invokes the PC Wrappers, they process the PIM interface following the FSM and control the pseudo-channels instead of the FPGA logic. The PC Wrappers deliver data configurations, such as input and output data addresses, to the pseudo-channels. We implement 32 KB of an SRAM buffer within each PC Wrapper for this purpose. As the PIM interface should be performed per pseudo-channel, we dedicate one PC Wrapper for each pseudo-channel. We connect a PC Wrapper with the pseudo-channel through an AXI interface. Then, the PC Wrapper informs the MPC Controller of the pseudo-channel's status.

D. Control Flow of MPC-Wrapper

Fig. 7 illustrates the control flow when the FPGA logic utilizes multiple pseudo-channels of Aquabolt-XL with MPC-Wrapper. In contrast to the complex procedure in the Section II-B, the FPGA logic can simply invoke MPC-Wrapper. In the preparation stage, MPC-Wrapper receives the level value through the `Level` port to allocate pseudo-channels to the PC set, and the MPC Controller configures the PC set accordingly. Subsequently, the MPC Controller receives instructions and data from the FPGA logic and performs the address partitioning logic. It deploys the divided instructions and data to PC Wrappers and updates the scoreboard to indicate whether the pseudo-channels of the PC set are busy. Then, the MPC Controller invokes the PC Wrappers belonging to the PC set to execute the PIM interface.

In the initialization stage, the PC Wrappers process the PIM interface. The initialization stage concludes with a mode switch to the AB-PIM mode and programming the CRF. The PC Wrappers proceed with the invocation stage, where the pseudo-channel performs the PIM operations by accessing data from the GRF or memory banks. After completing the operation, the PC Wrappers perform a mode switch, update the scoreboard to idle, and notify the MPC Controller of the termination in the finalization stage. Once the PIM interface of all PC Wrappers is finalized, the MPC Controller notifies the FPGA logic that the PIM operations have finished. Through this control flow, MPC-Wrapper effectively utilizes Aquabolt-XL's multiple pseudo-channels and relieves the FPGA logic from the burden of managing the complex low-level interactions. Also, MPC-Wrapper enables the FPGA logic to perform other FPGA tasks while carrying out the PIM interface.

E. Resource Sharing & Asynchronous Execution

MPC-Wrapper can concurrently accelerate different operations by resource sharing without performance degradation. For example, the FPGA logic intends to create two PC sets to comprise eight pseudo-channels of Aquabolt-XL separately. Then, the FPGA logic allocates the embedding lookup operation to one PC set and the GEMV operation to the other one while deploying the corresponding instructions and data to each PC set. Consequently, the pseudo-channels within each PC set execute the designated operations independently and in parallel. In other words, MPC-Wrapper's flexibility ensures applicability to utilize the pseudo-channels easily and the performance improvement by resource sharing.

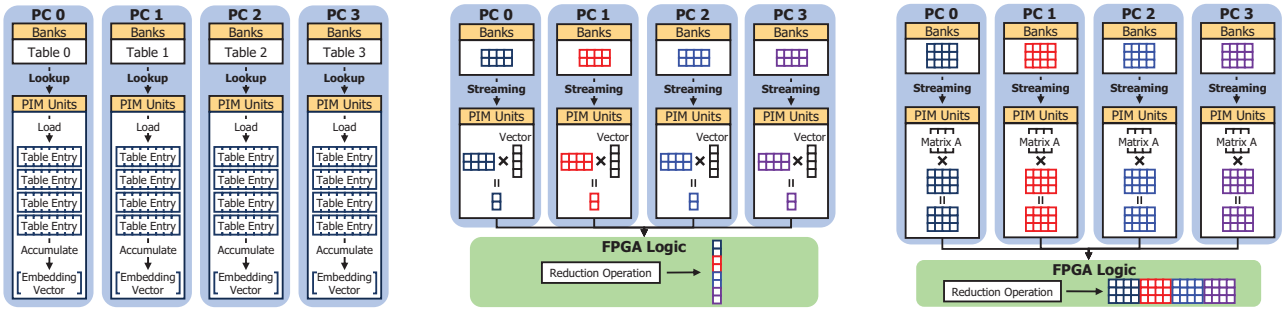
MPC-Wrapper enables asynchronous execution between the pseudo-channels and the FPGA logic by taking over the offloaded PIM interface from the FPGA logic. After the invocation of MPC-Wrapper for offloading an operation, the FPGA logic can execute a new operation immediately. Then, the execution of the pseudo-channels and the new operation of the FPGA logic are overlapped. Through the asynchronous execution, MPC-Wrapper can improve the overall performance as two operations are executed at the same time.

V. Multi-Pseudo-Channel Parallelization

We describe how memory-intensive benchmarks are implemented and parallelized with multiple pseudo-channels of Aquabolt-XL using MPC-Wrapper. Benchmarks comprise three key operations of deep learning models (i.e., embedding lookup, matrix-vector multiplication, and matrix-matrix multiplication) and an end-to-end deep learning model (i.e., Alibaba's recommendation model). We provide the practical implementation of the parallelization exploiting four pseudo-channels of Aquabolt-XL and MPC-Wrapper as an example.

A. Embedding Lookup

Embedding lookup is a memory-intensive and crucial operation in deep learning models. It generates an embedding vector



(a) Embedding lookup (b) GEMV (c) GEMM

Fig. 8: Multi-pseudo-channel parallelization of three memory-intensive benchmarks with MPC-Wrapper

by accumulating table entries from an embedding table. Since the operations for each embedding table are independent, we distribute embedding tables across multiple pseudo-channels and execute the operations in parallel. To fully exploit multiple pseudo-channels with MPC-Wrapper, we should implement and parallelize the operation by evenly distributing embedding tables across the pseudo-channels. Otherwise, some pseudo-channels must wait for others, which have more tables, to complete their operations. The load imbalance among the pseudo-channels makes Aquabolt-XL underutilized and execution latency longer. In addition, if we distribute a table across multiple pseudo-channels, the FPGA logic must load table entries from these pseudo-channels into the FPGA's buffer and sum them up, resulting in much higher latency.

Fig. 8a illustrates how we implement and parallelize the embedding lookup operation with MPC-Wrapper. The PC Wrappers deploy instructions to the pseudo-channels, directing the PIM units to look up the table based on the indices of table entries provided by the FPGA logic. Subsequently, the PIM units in each pseudo-channel load the table entries from their memory banks. The embedding vector is generated by accumulating these table entries and is written to the memory banks. Then, the FPGA logic loads the embedding vectors from the memory banks in each pseudo-channel. The operation finishes without any additional operations by the FPGA logic as the embedding tables are independent. Through this parallelization, we can effectively exploit all the potential of Aquabolt-XL and accelerate the operation.

B. Matrix-Vector Multiplication

General Matrix-Vector multiplication (GEMV) is a fundamental and extensively employed operation in deep learning models. To execute the GEMV operation across multiple pseudo-channels, it becomes imperative to distribute several matrix rows across the pseudo-channels. Leveraging the fact that generating a partial output vector from each matrix row is independent, multiple pseudo-channels can execute the partial GEMV operations independently and in parallel. For the same reason as the embedding lookup operation, we evenly distribute matrix rows to the pseudo-channels.

At the start of the operation, the PC Wrappers initially store an input vector into the GRF of each pseudo-channel, as shown in Fig. 8b. When the size of the input vector is larger than the capacity of the GRF, the PC Wrappers divide the input vector with respect to the capacity of the GRF. Then, the PC Wrappers deploy the divided input vectors to the pseudo-channel and execute the operation several times. The partial matrix is stored in the memory banks of the pseudo-channel. It is streamed into the PIM units and gets calculated. When the pseudo-channel completes its allocated partial GEMV operation, the partial output vector is stored in the memory banks. Due to the characteristics of Aquabolt-XL, it is infeasible for a pseudo-channel to access data of the other pseudo-channels, necessitating the FPGA logic to perform a reduction operation. The operation aggregates the partial output vectors to get the output vector. The FPGA logic loads these partial output vectors from the memory banks of the pseudo-channels into an FPGA's buffer and yields the output vector through the reduction operation.

C. Matrix-Matrix Multiplication

General Matrix-Matrix multiplication (GEMM) is a crucial operation in deep learning models to generate matrix C by multiplying matrix A with matrix B. To perform the GEMM operation using multiple pseudo-channels, the columns of matrix B need to be distributed across the pseudo-channels. Each pseudo-channel then carries out the matrix multiplication with matrix A and its allocated partial matrix of matrix B independently and in parallel. We ensure uniformity in the operation across pseudo-channels by distributing the same number of columns from matrix B.

The PC Wrappers store matrix A into the GRF of each pseudo-channel. When the size of matrix A is over the capacity of the SRAM buffer of the PC Wrapper, the FPGA logic has to divide matrix A to fit within the SRAM buffer and repeatedly invoke MPC-Wrapper with the partial matrices of A divided by rows. Making PC Wrapper's buffer larger mitigates extended execution latency by repeated invocations. However, this enhancement consumes more FPGA resources. Hence, a trade-off exists between the execution latency and the resource utilization. As illustrated in Fig. 8c, when starting the operation, the PIM units stream the partial matrix B from the memory banks and multiply two matrices. Then, they generate and store the partial output matrix in the memory banks. After

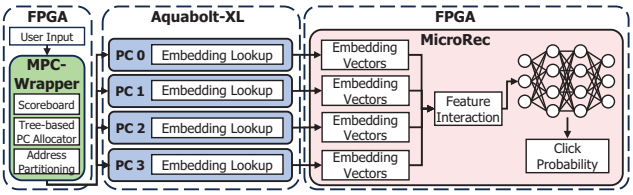


Fig. 9: Working model of AliRM [20] with MPC-Wrapper

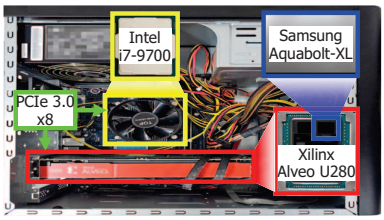


Fig. 10: Aquabolt-XL-equipped Xilinx Alveo U280 FPGA

all pseudo-channels complete their operations, the FPGA logic loads the partial output matrices from the memory banks of each pseudo-channel into the FPGA's buffer. Subsequently, it concludes the operation by generating the output matrix through the reduction operation.

D. Alibaba's Recommendation Model

To demonstrate MPC-Wrapper's ability to accelerate end-to-end deep learning models, we use Alibaba's recommendation model (AliRM) [20] and implement it using MicroRec [20], a representative FPGA-based recommendation model accelerator. AliRM is a memory-intensive recommendation model comprised of an embedding table lookup stage, feature interaction, and MLP layer executed in sequence. MicroRec is implemented to utilize all the available pseudo-channels of a standard HBM2 memory device. To utilize the pseudo-channels of Aquabolt-XL with MPC-Wrapper, we modify MicroRec to connect with Aquabolt-XL's pseudo-channels.

Fig. 9 shows a working model of AliRM with MPC-Wrapper with the FPGA logic. As the execution latency of the memory-intensive embedding table lookup stage occupies a large portion of the entire models' latency [20], [22], [23], we offload the embedding lookup to Aquabolt-XL. Initially, the FPGA logic checks which pseudo-channels are available using the scoreboard. It configures the PC set by sending the level value through the `Level` port to the MPC Controller. Following this, the FPGA logic provides the user input (i.e., the indices of embedding table entries) required for AliRM to MPC-Wrapper. The MPC Controller deploys the user input and data configuration, like the input and output addresses, to the PC Wrappers through the address partitioning logic. Subsequently, the PC Wrappers deliver them to the connected pseudo-channels and invoke the pseudo-channels to execute the embedding table lookup operation independently and in parallel. Once all the pseudo-channels complete the embedding lookup operation, the FPGA logic loads the embedding vectors from the pseudo-channels into the FPGA's buffer. The execution of AliRM gets completed by performing feature interaction and the MLP layer within the FPGA logic.

VI. Evaluation

A. Experimental Setup

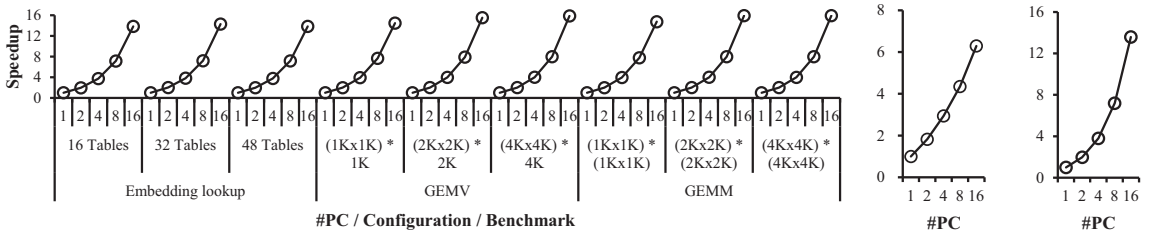
To evaluate MPC-Wrapper, we employ a PIM system equipped with an Intel i7-9700 CPU and an Aquabolt-XL-equipped Xilinx Alveo U280 FPGA [24], as shown in Fig. 10. The host CPU transfers data to Aquabolt-XL and controls the FPGA logic through PCIe 3.0 interface [25], and the FPGA logic exploits Aquabolt-XL through our MPC-Wrapper. We implement MPC-Wrapper using SystemVerilog and the FPGA logic using VHDL and High-Level Synthesis (HLS). We program the FPGA logic by Vivado 2023.1 [26]. We configure Aquabolt-XL to operate with the maximum frequency (i.e., 900 MHz [21]) and MPC-Wrapper to achieve an operating frequency of 300 MHz. We make the FPGA logic not only invoke MPC-Wrapper but also perform post-processing (e.g., reduction operation) by operating with the frequency of 300 MHz. To measure AliRM's latency, we employ the open-source MicroRec [20], which runs at 100 MHz. To assess the benefits of Aquabolt-XL compared to a standard HBM2 memory device, we implement the non-PIM implementations of the benchmarks of the standard HBM2 memory device.

Aimed at evaluating performance improvements and scalability of MPC-Wrapper, we execute the benchmarks with the baseline PIM implementations, which utilize only a single pseudo-channel of Aquabolt-XL. As benchmarks, we use three key operations of deep learning models (i.e., embedding lookup, GEMV, and GEMM) and AliRM [20] with a batch size of 32. Since Aquabolt-XL supports only the FP16 data type, we employ an FP16 synthetic dataset for the three key operations and a pre-trained FP16 dataset of MicroRec for AliRM. We examine three configurations each for key operations (e.g., 16, 32, and 48 tables for the embedding lookup operation). In addition, we use the configuration of AliRM as provided by its open-source implementation.

B. Large Speedup with MPC-Wrapper's Scalability

In this experiment, we evaluate the performance improvements and scalability of MPC-Wrapper. To perform an in-depth study on the speedups, we measure the execution latency of the PIM implementations of benchmarks using MPC-Wrapper, varying the number of pseudo-channels from 1 to 16. Fig. 11 depicts the speedups of PIM implementations over the baseline implementations. Overall, MPC-Wrapper achieves a geometric mean speedup of 13.66× over the baseline PIM implementation by employing all the 16 pseudo-channels. The results demonstrate the scalability of MPC-Wrapper, showing a linear increase in speedup with increased pseudo-channels.

MPC-Wrapper achieves geometric mean speedups of 13.98×, 15.27×, and 15.51× for the embedding lookup, GEMV, and GEMM operations, respectively, utilizing all the 16 pseudo-channels. The speedup for the embedding lookup operation is comparatively lower than those for the GEMV and GEMM operations. The reason is that the invocation



(a) Key operations of deep learning models (b) AliRM (c) Geomean

Fig. 11: Speedups of MPC-Wrapper with varying pseudo-channel counts

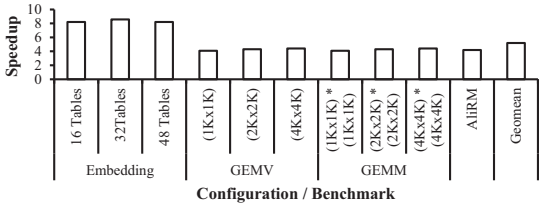
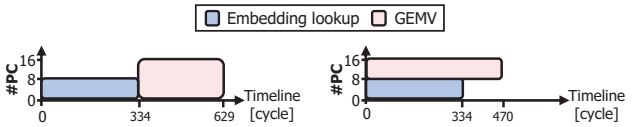
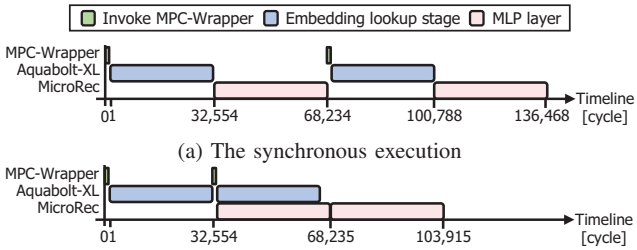


Fig. 12: Speedups of MPC-Wrapper over the baseline non-PIM implementations with 16 pseudo-channels



(a) Without resource sharing (b) With resource sharing

Fig. 13: Execution timelines of embedding lookup and GEMV without and with resource sharing



(a) The synchronous execution

(b) The asynchronous execution

Fig. 14: Execution timelines of two AliRM executions without and with asynchronous execution

stage is the only stage of the PIM interface affected by the speedup of multiple pseudo-channels. Since the invocation stage of the embedding lookup operation is shorter than those of other operations, it shows a relatively lower speedup. For AliRM, MPC-Wrapper achieves a speedup of 6.30×. The lower speedup, compared to the other benchmarks, is attributed to offloading only the embedding table lookup stage to Aquabolt-XL. In addition, despite being the same operation in Fig. 11a, the embedding table lookup stage of AliRM achieves a lower speedup of 12.10×. This is attributed to the fact that we implement the embedding lookup operation by equally distributing tables across the pseudo-channels. Because the embedding table lookup stage of AliRM includes 42 tables, the baseline PIM implementation processes these tables 42 times. Meanwhile, as the number of tables is not a multiple of the number of pseudo-channels, the PIM implementation with 16 pseudo-channels operates three times, leaving some pseudo-channels idle. Thus, the load imbalance among the pseudo-channels makes a lower speedup than depicted in Fig. 11a. Nevertheless, MPC-Wrapper gets notable speedups with all the benchmarks and ensures the scalability of Aquabolt-XL.

C. Speedup over the Non-PIM Implementation

To assess the benefits of Aquabolt-XL exploiting its full potential, we conduct a comparative analysis of execution latency in two scenarios: the PIM and the non-PIM implementations. Fig. 12 shows a geometric mean speedup of 5.19× achieved by MPC-Wrapper over the non-PIM implementation employing all the 16 pseudo-channels of the standard HBM2 memory device. The higher speedup of the embedding lookup operation is attributed to the fact that it accumulates the results in the GRF. In contrast, the GEMV and GEMM operations write back results to the memory banks when the size of the results exceeds the GRF's capacity.

D. Resource Sharing

The embedding operation involves eight tables, and the GEMV operation has a configuration of a 1K×128 matrix. Fig. 13 shows that resource sharing achieves a speedup of 1.34× compared to the latency without resource sharing. Without resource sharing, the two operations get executed in sequence, utilizing 8 and 16 pseudo-channels, respectively. This makes the embedding lookup operation leave the remaining eight pseudo-channels idle, as shown in Fig. 14a. On the other hand, the FPGA logic constructs two PC sets by resource sharing of MPC-Wrapper, each consisting of eight pseudo-channels. Resource sharing improves the overall performance and the utilization of Aquabolt-XL by allocating the embedding lookup operation and the GEMV operation to the respective PC sets and concurrently executing them.

E. Asynchronous Execution

Fig. 14 illustrates the execution timelines for an example scenario where two AliRMs are performed synchronously and asynchronously on Aquabolt-XL and MicroRec with MPC-Wrapper. With the enhanced usability by MPC-Wrapper, the asynchronous execution between Aquabolt-XL and the MicroRec achieves a speedup of 1.31× compared to the

TABLE I: Resource utilization of MPC-Wrapper on the Aquabolt-XL-equipped Xilinx Alveo U280 FPGA

Component	LUT	Register	BRAM
MPC Controller	8,211 (0.63%)	14,416 (0.55%)	0 (0%)
PC Wrappers	51,349 (3.94%)	68,368 (2.62%)	120 (5.95%)
Total (MPC-Wrapper)	59,560 (4.57%)	82,784 (3.18%)	120 (5.95%)

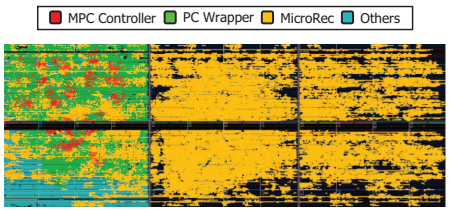


Fig. 15: Floorplan of MPC-Wrapper and MicroRec on the Aquabolt-XL-equipped Xilinx Alveo U280 FPGA

synchronous execution. In the synchronous execution, we invoke MPC-Wrapper for the start of the second AliRM after the completion of the first AliRM execution. In contrast, the asynchronous execution by MPC-Wrapper ensures an overlap of the MLP layer of the first AliRM on MicroRec with the embedding table lookup stage of the second AliRM on Aquabolt-XL. The MLP layer of the first AliRM gets delayed by only one cycle due to the invocation of MPC-Wrapper to offload the second AliRM to Aquabolt-XL, which has a negligible impact on the overall performance.

F. Hardware Implementation Costs

Table I shows the resource utilization of MPC-Wrapper, implemented on the Aquabolt-XL-equipped Xilinx Alveo U280 FPGA using all 16 pseudo-channels. MPC-Wrapper demands only a minimal amount of the FPGA resources. Although the MPC Controller supports features such as the scoreboard, the tree-based PC allocator, and the address partitioning logic, it utilizes only slight resources. Similarly, the PC Wrappers, which employ all the 16 pseudo-channels, also consume small resources. They use LUTs and registers to execute the FSM logic for the PIM interface and consume BRAMs less than 6% to store data deployed to the dedicated pseudo-channel. With the addition of Aquabolt-XL equipment, although MPC-Wrapper's resource utilization might increase due to the MPC Controller expansion and the scaling out of the PC Wrappers and `PC` ports, we anticipate the increase to remain insignificant. In addition, as shown in Fig. 15, the floorplan comparison between MPC-Wrapper and MicroRec shows that MPC-Wrapper requires considerably fewer resources than MicroRec, which utilizes 26.60%, 8.23%, and 92.76% of LUTs, registers, and BRAMs, respectively. It demonstrates that MPC-Wrapper can be seamlessly integrated with any FPGA logic (e.g., MicroRec) thanks to its slight resource usage.

VII. Related Work

A. Commodity Processing-In-Memory Devices

UPMEM [14] and AxDIMM [27] are PIM-enabled DIMM devices which place IMPs near memory banks or ranks. They are compatible with the DIMM slots available on real systems and can be used as a drop-in replacement for standard DIMMs. Unlike Aquabolt-XL, the PIM-enabled DIMM devices require byte-interleaving, leading to additional efforts for accurate data usage by IMPs due to the DIMM's characteristics. In addition, AiM [15], which places a multiply-accumulate unit adjacent to a memory bank, stands out as a representative PIM-enabled GDDR6-based memory device that can function as both a PIM device and a standard memory. However, AiM necessitates the use of additional DRAM commands and requires modifications to the host-side memory controller.

B. Leveraging Aquabolt-XL for Acceleration

Kang et al. [18] accelerate RNN-T [19], a translation model, using an Aquabolt-XL-equipped Xilinx Alveo U280 FPGA. Although they utilize only a single pseudo-channel of Aquabolt-XL, they are able to achieve significant performance improvements by taking advantage of Aquabolt-XL, which significantly reduce the data movement between memory devices and computational units. We believe that RNN-T can achieve more performance improvement by leveraging all the 16 pseudo-channels using MPC-Wrapper. Moreover, Kim et al. [28] study on GPUs equipped with Aquabolt-XL. They demonstrate a significant overall performance improvement when GPT-J is executed on a single Aquabolt-XL-equipped AMD MI100 GPU compared to an AMD MI100 GPU equipped with a standard HBM2 memory device. They also build an Aquabolt-XL cluster, installing 96 Aquabolt-XL-equipped AMD MI100 GPUs. Executing a Transformer-based mixture of experts model on the Aquabolt-XL-equipped GPU cluster results in a notable speedup compared to a non-PIM GPU cluster. Unlike this work, they do not discuss the multiple pseudo-channels of Aquabolt-XL and study on an FPGA.

VIII. Conclusion

We proposed MPC-Wrapper, a first novel approach designed to fully harness the potential of Samsung Aquabolt-XL by effectively utilizing all the PIM-enabled pseudo-channels. Through the implementation of the MPC Controller, MPC-Wrapper ensures scalability by enabling the parallel execution of the pseudo-channels. The MPC Controller further provides flexibility by exposing all the pseudo-channels as separate ports, allowing the PC set to be configured dynamically. In addition, MPC-Wrapper's PC Wrappers enhance usability by offloading the PIM interface from the FPGA logic and managing the pseudo-channels. Our experimental results show that MPC-Wrapper achieves a notable speedup of 13.66× over the baseline single-pseudo-channel PIM implementations.

Acknowledgment

This work was supported by Samsung Advanced Institute of Technology (SAIT). This work was also supported in part by the National Research Foundation of Korea (NRF) under Grant 2022R1C1C1008131 and the Institute of Information & Communications Technology Planning & Evaluation (IITP) under Grant 2020-0-01361.

REFERENCES

[1] M. Naumov, D. Mudigere, H.-J. M. Shi, J. Huang, N. Sundaraman, J. Park, X. Wang, U. Gupta, C.-J. Wu, A. G. Azzolini, *et al.*, "Deep Learning Recommendation Model for Personalization and Recommendation Systems," *arXiv preprint arXiv:1906.00091*, 2019.

[2] W. Fedus, B. Zoph, and N. Shazeer, "Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity," *The Journal of Machine Learning Research*, vol. 23, 2022.

[3] S. H. Pugsley, J. Jestes, H. Zhang, R. Balasubramonian, V. Srinivasan, A. Buyuktosunoglu, A. Davis, and F. Li, "NDC: Analyzing the Impact of 3D-Stacked Memory+Logic Devices on MapReduce Workloads," in *Proc. 2014 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, 2014.

[4] J. Ahn, S. Hong, S. Yoo, O. Mutlu, and K. Choi, "A Scalable Processing-in-Memory Accelerator for Parallel Graph Processing," in *Proc. 42nd IEEE/ACM International Symposium on Computer Architecture (ISCA)*, 2015.

[5] C. Lim, S. Lee, J. Choi, J. Lee, S. Park, H. Kim, J. Lee, and Y. Kim, "Design and Analysis of a Processing-in-DIMM Join Algorithm: A Case Study with UPMEM DIMMs," *Proceedings of the ACM on Management of Data (PACMMOD)*, vol. 1, 2023.

[6] M. Gao, J. Pu, X. Yang, M. Horowitz, and C. Kozyrakis, "Tetris: Scalable and Efficient Neural Network Acceleration with 3D Memory," in *Proc. 22nd ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2017.

[7] M. Peemen, A. A. A. Setio, B. Mesman, and H. Corporaal, "Memory-Centric Accelerator Design for Convolutional Neural Networks," in *Proc. 31st IEEE International Conference on Computer Design (ICCD)*, 2013.

[8] L. Nai, R. Hadidi, J. Sim, H. Kim, P. Kumar, and H. Kim, "GraphPIM: Enabling Instruction-Level PIM Offloading in Graph Computing Frameworks," in *Proc. 23rd IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, 2017.

[9] Y. Kwon, Y. Lee, and M. Rhu, "TensorDIMM: A Practical Near-Memory Processing Architecture for Embeddings and Tensor Operations in Deep Learning," in *Proc. 52nd IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2019.

[10] 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 *Proc. 47th IEEE/ACM International Symposium on Computer Architecture (ISCA)*, 2020.

[11] C. Xie, S. L. Song, J. Wang, W. Zhang, and X. Fu, "Processing-in-Memory Enabled Graphics Processors for 3D Rendering," in *Proc. 23rd IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, 2017.

[12] C. Xie, X. Zhang, A. Li, X. Fu, and S. Song, "PIM-VR: Erasing Motion Anomalies In Highly-Interactive Virtual Reality World with Customized Memory Cube," in *Proc. 25th IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, 2019.

[13] C. Xie, X. Li, Y. Hu, H. Peng, M. Taylor, and S. L. Song, "Q-VR: System-Level Design for Future Mobile Collaborative Virtual Reality," in *Proc. 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2021.

[14] F. Devaux, "The true Processing In Memory accelerator," in *Proc. 31th IEEE Hot Chips Symposium (HCS)*, 2019.

[15] M. He, C. Song, I. Kim, C. Jeong, S. Kim, I. Park, M. Thottethodi, and T. Vijaykumar, "Newton: A DRAM-maker's Accelerator-in-Memory (AiM) Architecture for Machine Learning," in *Proc. 53rd IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2020.

[16] S. Lee, S.-h. Kang, J. Lee, H. Kim, E. Lee, S. Seo, H. Yoon, S. Lee, K. Lim, H. Shin, J. Kim, S. O, A. Iyer, D. Wang, K. Sohn, and N. S. Kim, "Hardware Architecture and Software Stack for PIM Based on Commercial DRAM Technology," in *Proc. 48th IEEE/ACM International Symposium on Computer Architecture (ISCA)*, 2021.

[17] J. H. Kim, S.-H. Kang, S. Lee, H. Kim, Y. Ro, S. Lee, D. Wang, J. Choi, J. So, Y. Cho, J. Song, J. Cho, K. Sohn, and N. S. Kim, "Aquabolt-XL HBM2-PIM, LPDDR5-PIM With In-Memory Processing, and AXDIMM With Acceleration Buffer," *IEEE Micro*, vol. 42, 2022.

[18] S. Kang, S. Lee, B. Kim, H. Kim, K. Sohn, N. S. Kim, and E. Lee, "An FPGA-based RNN-T Inference Accelerator with PIM-HBM," in *Proc. 30th ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA)*, 2022.

[19] Y. He, T. N. Sainath, R. Prabhavalkar, I. McGraw, R. Alvarez, D. Zhao, D. Rybach, A. Kannan, Y. Wu, R. Pang, *et al.*, "Streaming End-To-End Speech Recognition for Mobile Devices," in *Proc. 44th IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019.

[20] W. Jiang, Z. He, S. Zhang, T. B. Preußer, K. Zeng, L. Feng, J. Zhang, T. Liu, Y. Li, J. Zhou, *et al.*, "MicroRec: Efficient Recommendation Inference by Hardware and Data Structure Solutions," *Proceedings of Machine Learning and Systems (MLSys)*, vol. 3, 2021.

[21] Xilinx, Inc., "AXI High Bandwidth Memory Controller LogiCORE IP Product Guide (PG276)," 2022. https://docs.amd.com/r/en-US/pg276-axi-hbm.

[22] L. Ke, U. Gupta, B. Y. Cho, D. Brooks, V. Chandra, U. Diril, A. Firoozshahian, K. Hazelwood, B. Jia, H.-H. S. Lee, *et al.*, "RecNMP: Accelerating Personalized Recommendation with Near-Memory Processing," in *Proc. 47th IEEE/ACM International Symposium on Computer Architecture (ISCA)*, 2020.

[23] R. Jain, S. Cheng, V. Kalagi, V. Sanghavi, S. Kaul, M. Arunachalam, K. Maeng, A. Jog, A. Sivasubramaniam, M. T. Kandemir, *et al.*, "Optimizing CPU Performance for Recommendation Systems At-Scale," in *Proc. 50th IEEE/ACM International Symposium on Computer Architecture (ISCA)*, 2023.

[24] Xilinx, Inc., "Alveo U280 Data Center Accelerator Card User Guide (UG1314)," 2023. https://docs.amd.com/r/en-US/ug1314-alveo-u280-reconfig-accel.

[25] J. Ajanovic, "PCI Express 3.0 Overview," in *Proc. 21th IEEE Hot Chips Symposium (HCS)*, 2009.

[26] Xilinx, Inc., "Vivado Design Suite (WP416)," 2012. https://docs.amd.com/v/u/en-US/wp416-Vivado-Design-Suite.

[27] L. Ke, X. Zhang, J. So, J.-G. Lee, S.-H. Kang, S. Lee, S. Han, Y. Cho, J. H. Kim, Y. Kwon, *et al.*, "Near-memory Processing in Action: Accelerating Personalized Recommendation with AxDIMM," *IEEE Micro*, vol. 42, 2021.

[28] J. H. Kim, Y. Ro, J. So, S. Lee, S.-h. Kang, Y. Cho, H. Kim, B. Kim, K. Kim, S. Park, *et al.*, "Samsung PIM/PNM for Transformer Based AI: Energy Efficiency on PIM/PNM Cluster," in *Proc. 35th IEEE Hot Chips Symposium (HCS)*, 2023.