Minimizing Communication Conflicts in Network-On-Chip Based Processing-In-Memory Architecture

Hanbo Sun[1], Tongxin Xie[1], Zhenhua Zhu[1], Guohao Dai[2], Huazhong Yang[1], Yu Wang[1†]
[1] Department of Electronic Engineering, BNRist, Tsinghua University, Beijing, China
[2] Qing Yuan Research Institute, Shanghai Jiao Tong University, Shanghai, China

Abstract*—Deep Neural Networks (DNNs) have made significant breakthroughs in various fields. However, their enormous computations and parameters seriously hinder their applications. Emerging Processing-In-Memory (PIM) architectures provide extremely high energy efficiency to accelerate DNN computing. Moreover, Network-on-Chip (NoC) based PIM architectures significantly improve the scalability of PIM architectures. However, the contradiction between high communication and limited NoC bandwidth introduces severe communication conflicts. Existing work neglects the impact of communication conflicts. On the one hand, neglecting communication conflicts leads to the lack of precise performance estimations in the mapping process, making it hard to find optimal results. On the other hand, communication conflicts cause low NoC bandwidth utilization in the schedule process. And there is over 70% latency gap in existing work caused by communication conflicts. This paper proposes communication conflict optimized mapping and schedule strategies for NoC-based PIM architectures. The proposed mapping strategy constructs communication conflict graphs to model communication conflicts. Based on this constructed graph, we adopt a Graph Neural Network (GNN) as a precise performance estimator. Our schedule strategy predefines the communication priority and NoC communication behavior tables for target DNN workloads. In this way, it can improve the NoC bandwidth utilization effectively. Compared with existing work, for typical classification DNNs on the CIFAR and ImageNet datasets, the proposed strategies reduce 78% latency and improve the throughput by 3.33× on average with negligible deployment and hardware overhead. Experimental results also show that our strategies decrease the average gap to ideal cases without communication conflicts from 80.7% and 70% to 12.3% and 1.26% for latency and throughput, respectively.

I. Introduction

In recent years, Deep Neural Networks (DNNs) have made breakthroughs in various fields, such as computer vision [1] and natural language processing [2]. However, DNNs introduce massive amounts of computations and parameters, making DNN computing both time and energy consuming.

Based on the emerging Non-Volatile Memories (NVMs), researchers have proposed Processing-In-Memory (PIM) architectures to accelerate DNN computing [3] [4]. PIM architectures perform *in-situ* Matrix-Vector-Multiplications (MVMs) in the memory, avoiding redundant data movement. Therefore, PIM architectures can improve energy efficiency by three orders of magnitude over GPU and CMOS ASIC solutions and show great potential to accelerate DNN computing [3] [4].

The basic computing components in PIM architectures are called as Processing Elements (PEs). Each PE is composed of crossbars, i.e., NVM cells organized in an array form, and peripheral circuits. Limited by the crossbar size, multiple PEs are needed to deploy DNNs. For instance, the parameter amounts of VGG-8 [5] and ResNet-18 [6] are around 9.3×10^6 and 1.4×10^7, respectively. And for typical PEs with four 256×256 crossbars, 61 and 120 PEs are required to deploy them. To manage the computations and communication among hundreds of PEs, Network-On-Chip (NoC) based PIM architectures [7] have been proposed. Compared with Point-to-Point interconnect based PIM architectures, NoC-based PIM architectures can achieve better scalability and realize up to $14.94\times$ throughput improvement [8].

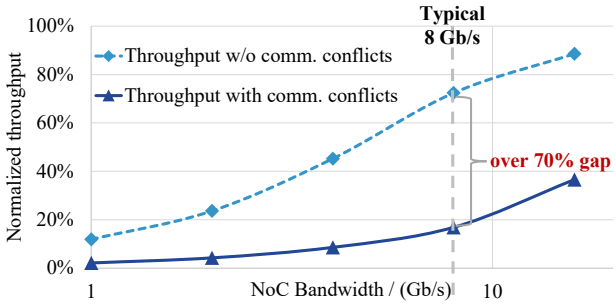

Fig. 1. Comparison of the throughput with and w/o communication conflicts of ResNet-18 [6] on the CIFAR [9] dataset under different NoC bandwidth. The hardware is the typical NoC-based PIM architecture in MNSIM 2.0 [10].

However, communication conflicts [11] have become the critical bottleneck in NoC-based PIM architectures. In NoC-based PIM architectures, different PEs perform computations in parallel, generating output data to be transmitted concurrently. Therefore, communication requests from different PEs are highly concurrent. For example, the number of communication requests at the same time can reach up to 101 and 192 for VGG-8 and ResNet-18, respectively. Highly concurrent communication requests bring severe communication conflicts. Communication conflicts block the data movement and harm the latency and throughput. As shown in Figure 1, we compare the realistic throughput with the throughput without communication conflicts. The ideal cases without communication conflicts suppose that all communication requests can be executed immediately. Figure 1 shows communication conflicts reduce the throughput by over 70% under the typical NoC bandwidth.

The flow of deploying DNNs in NoC-based PIM architectures consists of mapping and schedule processes. The mapping process assigns different parts of DNN workloads to different PEs offline. The schedule process assigns communication resources, e.g., the NoC communication path, to communication tasks during runtime. Researchers have designed several mapping and schedule strategies for NoC-based PIM architectures [12]–[16]. However, communication conflicts make existing mapping and schedule strategies ineffective. On the one hand, in the mapping process, communication conflicts make it difficult to estimate performance, e.g., the latency and throughput. And the lack of precise performance estimations makes it hard to find optimal results. For example, existing mapping strategies adopt total communication cost, i.e., the sum of communication amount multiplied by the communication

†: Corresponding author (yu-wang@tsinghua.edu.cn)

978-3-9819263-7-8/DATE23/© 2023 EDAA

distance, to estimate the latency [15] [16]. And according to our experimental results, the correlation coefficient between the total communication cost and realistic latency is lower than 0.1. Therefore, the optimal total communication cost does not mean the optimal latency, causing around 35% latency increment.

On the other hand, in the schedule process, strict data dependencies of DNNs cause severe communication bandwidth underutilization. The computations and communication of DNNs must execute in the algorithm-specific order to ensure correct results. The algorithm-specific order results in strict data dependencies among PEs. Communication conflicts and strict data dependencies block the subsequent computations. Thus, the subsequent computations can not produce new data to be transmitted, wasting communication resources. We find that over 56% communication resources have a NoC bandwidth utilization rate lower than 10%, and the average NoC bandwidth utilization rate is only 11%.

To tackle these problems, we propose communication conflict optimized mapping and schedule strategies for NoC-based PIM architectures. The main contributions of this paper contain:

- In the mapping process, we introduce communication conflict graphs to model communication conflicts. Furthermore, we construct a Graph Neural Network (GNN) based latency and throughput estimator. The GNN based estimator raises the correlation coefficient between the estimated performance and the realistic performance from 0.01 to 0.78.
- In the schedule process, we propose a workload balance arbiter and a behavior table based NoC routing algorithm in the schedule process. The corresponding router structure is also designed to support multi-communication pipeline execution. Therefore, our schedule strategy improves the average NoC bandwidth utilization rate from 11.2% to 53.5% ($\sim$4.78$\times$).
- Extensive experimental results show that, for typical classification DNNs on the CIFAR [9] and ImageNet [1] datasets, the proposed mapping and schedule strategies can reduce 78% latency and improve throughput by $3.33\times$ on average. Compared with the ideal cases without communication conflicts, the proposed strategies reduce the average gaps from 80.7% and 70% to 12% and 1.26% for the latency and throughput, respectively.

II. Background

A. DNN Representation

We use Directed Acyclic Graphs (DAGs) to represent DNNs. In DAGs, each vertex v_i denotes a DNN operation, e.g., the convolutional or linear operation. And the edge from v_i to v_j represents the data flow from the operation v_i to the operation v_j. The operations and data flows must be executed in the algorithm-specific order to produce the right results.

B. NoC-based PIM Architecture

As shown in the right part of Figure 2, NoC-based PIM architectures contain multiple PEs, Network Interfaces (NIs), routers, and interconnection wires among routers. And each PE has a private NI and a bound router. In NoC-based PIM architectures, we can express computation resources as follows:

$$CR = \{PE_1, PE_2 \cdots, PE_M\} \quad (1)$$

where M denotes the total number of PEs.

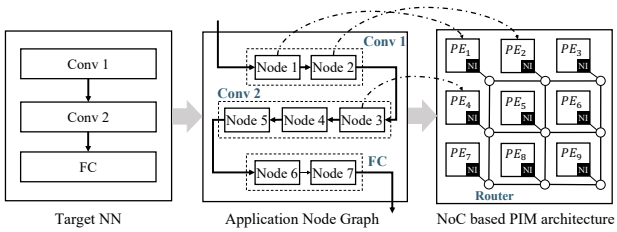

Fig. 2. Basic mapping flow of deploying DNNs in PIM architectures.

C. Related Work

Krishnan *et al.* [12] and Mandal *et al.* [13] reconfigure NoC topology to relieve communication conflicts. They design different customized hardware for different target DNNs. However, customized designs need a long production period and are difficult to adapt to the rapidly evolving DNNs.

MNSIM 2.0 [10] and Krishnan *et al.* [8] utilize workload-independent mapping and schedule methods for general NoC-based PIM architectures, e.g., row-by-row mapping. And Zhou *et al.* [14], Liu *et al.* [15], and Ji *et al.* [16] propose workload-aware mapping and schedule strategies. Liu *et al.* [15] utilizes a heuristic algorithm to find optimal mapping results, and Zhou *et al.* [14] generate computation and communication instructions to schedule computations and communication among different PEs. However, these mapping and schedule strategies lack consideration of communication conflicts, leading to the lack of precise performance estimations in the mapping process and low NoC bandwidth utilization in the schedule process. Maqsood *et al.* [11] perform workload balancing on NoC links to reduce communication conflicts. However, they do not consider global communication conflicts.

III. Mapping Strategy

A. Fundamental Mapping Flow

Figure 2 shows the basic mapping flow of deploying DNNs in NoC-based PIM architectures. The mapping flow consists of three main components: target DNNs as the input, mapping results in PIM architectures as the output, and Application Node Graphs (ANGs) as the Intermediate Representation (IR).

For specified NoC-based PIM architectures, each PE's storage and computation capability are invariant. For DNNs, the number of parameters and computations changes with different hyperparameters, e.g., different kernel sizes. To address this contradiction, we introduce ANGs as the IR between operations and PEs. Each node in ANGs is a part of an operation and can be deployed in a PE. And edges correspond to the data dependencies among nodes. The mapping process is converted to assigning nodes in ANGs to PEs in PIM architectures:

$$Mapping(ANG, \{PE_i\}) = \{n_i \rightarrow PE_k | n_i \in ANG\} \quad (2)$$

In this paper, we propose a communication conflict optimized mapping strategy for NoC-based PIM architectures, which is shown in Figure 3. Our mapping strategy consists of a GNN based estimator to precisely estimate latency and throughput and a node grouping algorithm to reduce the search space size. Besides, we adopt a heuristic algorithm [17] to search for optimal mapping results.

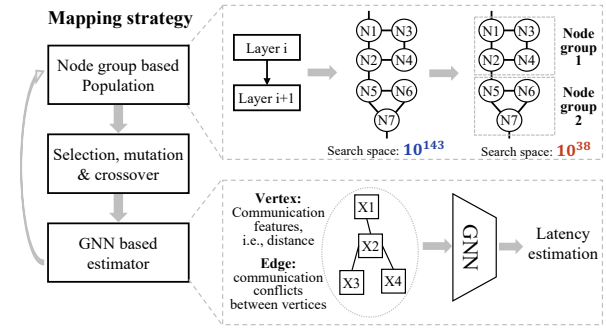

Fig. 3. Proposed mapping strategy.

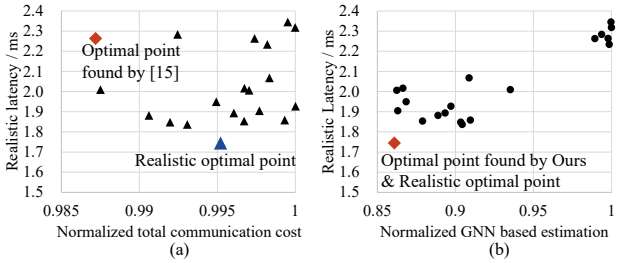

Fig. 4. Correlation comparison of (a) total communication cost and (b) GNN based estimation results with realistic results in VGG-8 mapping results.

B. GNN based Estimator

As mentioned in Section I, the lack of precise performance estimations makes it hard to find optimal results. For instance, existing mapping methods [8] [15] adopt the total communication cost as the latency and throughput estimation. As shown in Figure 4 (a), the x-axis denotes the normalized total communication cost and the y-axis stands for the realistic latency. These mapping methods will find the "red" point as the "optimal" result. And there is ∼35% latency gap between the found optimal point and the realistic optimal point.

Poor performance estimation is caused by neglecting communication conflicts. As shown in Table I, neglecting communication conflicts decreases the correlation coefficient from 0.67 to 0.24 for the Multi Layer Perceptron (MLP) based estimator. Communication conflicts mean different communication tasks requesting the same communication resources. In other words, communication conflicts denote the relationship of the competition for the bandwidth among different communication tasks. This kind of relationship is sparse, which is unsuitable to be characterized by the dense form, e.g., vectors and matrices. For sparse relationships, graphs have been widely used as the data structure [18]. Inspired by this, we adopt graphs to model communication conflicts, named communication conflict graphs.

In communication conflict graphs, each vertex stands for one communication task, and vertex features consist of communication data amounts, communication distance, and other essential information about this communication task. As for the edges in communication conflict graphs, if and only if there are communication conflicts between the i_{th} and the j_{th} communication tasks, i.e., v_i and v_j, there will be a corresponding edge $e_{i,j}$ between v_i and v_j.

Afterward, we set up a Graph Neural Network (GNN) based estimator [18]. As shown in Table I, owing to the communication conflict graphs, the GNN based estimator can

TABLE I
CORRELATION COEFFICIENT UNDER DIFFERENT ESTIMATION METHODS

Estimation Method	Total comm. cost	MLP w/o comm. conflicts	MLP with comm. conflicts	GNN estimator
Correlation coef.	0.01	0.24	0.67	**0.78**

effectively handle communication conflict information, and obtains the best correlation coefficient. Compared with vector inputs with communication conflicts in MLP estimators, the communication conflict graphs further improve the correlation coefficient by around 11%. Figure 4 (b) shows that the GNN based estimator can find the realistic "optimal" point.

C. Node Grouping Algorithm

As shown in Figure 2, every node in ANGs can be assigned to any PE in the PIM array, leading to vast search space in the mapping process. For instance, when mapping 61 nodes of VGG-8 in a 16×16 PE array, the search space size reaches 3.3×10^{143}. A vast search space leads to a long search time and makes it hard to find optimal mapping results.

To narrow the search space, we propose a node grouping algorithm. The node grouping algorithm narrows the search space by decreasing the number of searchable variables. In our algorithm, we introduce node groups, and each node group refers to a set of nodes whose relative positions can be determined. We only need to assign positions to each node group in our algorithm. Since the number of node groups is much smaller than all nodes, the node grouping algorithm can significantly reduce the search space size. Experimental results show that our algorithm can reduce the search space size from 3.3×10^{143} to 2.5×10^{38}. Besides, the heuristic algorithm with the node grouping algorithm needs only one-third of iterations to acquire comparable mapping results.

To determine node groups in ANGs, we partition and group all nodes based on the communication amount among nodes. Firstly, we count the communication amount between every two nodes and sort them in descending order. Secondly, we analyze every two nodes: If neither node is in existing node groups, these two nodes construct a new node group. And if only one node is in existing node groups, this node group adds the other node. Finally, for each node group, the relative positions are determined by minimizing the communication distance of every two nodes inside the node group. The node grouping algorithm splits the node position assignment into two stages: relative position assignment inside node groups and node group position assignment. By partitioning node groups based on the communication amounts, the node group algorithm ensures stable mapping results to a large extent.

Algorithm 1 shows the details of the proposed mapping algorithm. The inputs are the hardware description and ANGs, and the outputs are the mapping results. The proposed mapping algorithm consists of three main stages. In the first stage, node groups are constructed in the traversal of the sorted list of ANGs (**line** 1∼8). Secondly, we initialize random mapping results based on the node groups, named $MappingResults$. Then, for each mapping candidate in the $MappingResults$, we get the hardware performance as $Target$, and the input

Algorithm 1: Pseudocode of our mapping strategy.

Input: Hardware: $CR = \{PE_1, PE_2 \cdots, PE_M\}$;
Application Node Graph: $ANG = G(N, E)$.
Output: $MappingResults = Mapping(ANG, CR)$

```
1  Partition nodes into groups:
2  Sort edges List = Sort(E(v_i, v_j));
3  for E(v_i, v_j) ∈ List do
4      if neither in existing node groups then
5          Construct new node group;
6      else if one in existing node groups then
7          Add the other to this node group;
8  end
9  Train the GNN based estimator:
10 Random init. based on node groups MappingResults;
11 for MappingCand ∈ MappingResults do
12     Target = performance of MappingCand;
13     Input = input graph of MappingCand;
14     Train(GNN, Target, Input);
15 end
16 Heuristic search with specified node groups and GNN;
```

communication conflict graph as $Input$, based on the simulation results. Afterward, we train the GNN based estimator with the $Target$ and the $Input$ (**line** 14). Finally, a heuristic search is performed to find optimal mapping results (**line** 16).

IV. Schedule Strategy

As mentioned in Section I, the communication conflicts and strict data dependencies of DNNs cause severe NoC bandwidth underutilization. To enhance the bandwidth utilization, we propose a communication conflict optimized schedule strategy, consisting of a workload balance arbiter and a behavior table based NoC routing algorithm.

A. Workload Balance Arbiter

Arbiters are devices that arbitrate communication requests when a communication conflict happens. Existing arbiters contain fixed priority arbiters [19] and dynamic adaptive arbiters [20]. The fixed priority arbiters are easy to implement but lead to low NoC bandwidth utilization. The dynamic adaptive arbiters improve the NoC bandwidth utilization by changing priority dynamically, but may cause some data not to reach target PEs, e.g., live lock. Moreover, both fixed priority and dynamic adaptive arbiters lack considerations of strict data dependencies of DNNs, making communication conflicts block the subsequent computations. As a result, the subsequent computations can not produce new data, wasting communication bandwidth resources.

To address this problem, we propose a workload balance arbiter to consider the strict data dependencies of DNNs. The workload balance arbiter assigns the priority based on the transmission rate of communication tasks, which can be described as follows:

$$Priority = \frac{Data_{total}}{Data_{trans}} \tag{3}$$

where $Data_{total}$ and $Data_{trans}$ denote the amount of total data and transferred data of communication tasks, respectively. The more data has been transferred, the lower priority is assigned to the communication task. Considering that data dependencies occur in a near proportion of different communication tasks, our schedule method can process communication-heavy tasks efficiently. As shown in Figure 5, the workload balance arbiter enables communication tasks to be executed in pipeline mode. Experimental results show that, for typical classification DNNs on the CIFAR and ImageNet datasets, the workload balance arbiter reduces the average latency by around 30%.

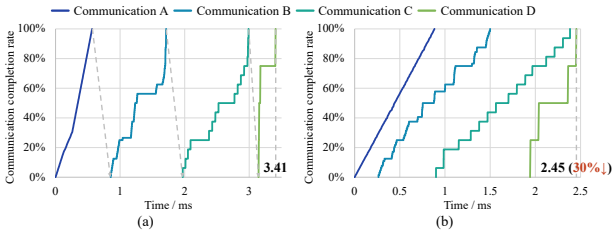

Fig. 5. Execution process of the same communication tasks under (a) a fixed priority arbiter and (b) the proposed workload balance arbiter.

B. Behavior Table based Routing Algorithm

Besides the workload balance arbiter, we propose a behavior table based NoC routing algorithm. The behavior table based NoC routing algorithm plugs a behavior table into each router. The data in the behavior table is in the following form:

$$[O_1, O_2 \cdots, O_K], O_i : Start_i \rightarrow End_i \tag{4}$$

where O_i denotes the i_{th} operation data in the behavior table, and $Start_i$ and End_i represent the source and target orientation, respectively. Source and target orientations are in North, South, West, East, and the PE. For example, $O_1 : PE \rightarrow North$ means the first operation is transferring data generated by this PE to the adjacent north router. Appropriate behavior tables can realize high NoC bandwidth utilization and perform DNN workloads efficiently.

To generate behavior tables for one communication task, we propose an extended Dijkstra algorithm [21] as follows:

$$Path(S,T,G) = \begin{cases} None & \text{if Dis(D(S,T,G))} > \text{R*M(S,T)} \\ D(S,T,G) & \text{if Dis(D(S,T,G))} \leq \text{R*M(S,T)} \end{cases} \tag{5}$$

where $D(S, T, G)$ returns the path that the Dijkstra algorithm finds from S as the source vertex to T as the target vertex in graph G. $Dis(\cdot)$ denotes the function to get the distance of the path. $M(S, T)$ denotes the Manhattan distance from S to T, and R represents the hyperparameter in our algorithm. When the path given by the Dijkstra algorithm is too long (the first line in Equation 5), we choose to wait for a shorter path rather than spend a long communication time. G is initialized by the NoC topology, and the edge denotes the link among routers. Then according to the communication start time, we perform the extended Dijkstra algorithm for every communication task. When any communication tasks start or end, we update the edges of the graph G, i.e., removing or adding corresponding edges. Finally, we obtain all the data in all behavior tables.

As shown in Figure 6, the proposed routing algorithm can consistently achieve lower latency than the XY routing algorithm [22]. Compared with 1 and 10, $R = 2$ ensures

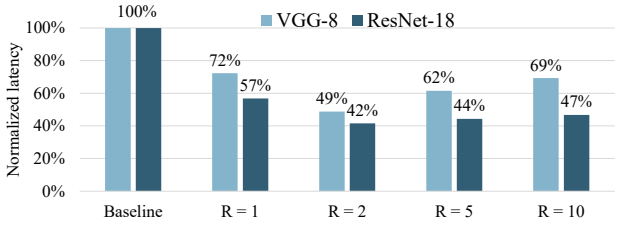

Fig. 6. Latency under different R values in the proposed extended Dijkstra routing algorithm, and the baseline is the XY routing algorithm.

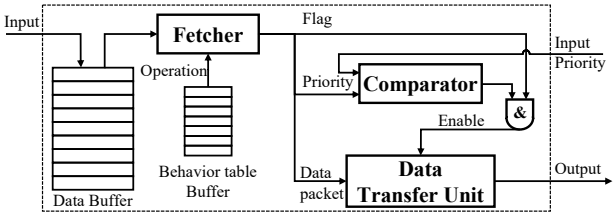

Fig. 7. Router structure design to support the proposed schedule strategy.

full exploration of the communication path, avoids too long communication paths, and achieves the lowest latency.

Our schedule strategy can maximize the communication parallelism with satisfying the strict data dependencies. Experimental results show our schedule strategy improves the average NoC bandwidth utilization rate from 11.2% to 53.5%. Considering the inherent workload imbalance among different operations of DNNs [23], there is an upper bound for the NoC bandwidth utilization. In the ideal scenario without communication conflicts, the NoC bandwidth utilization rate reaches 54.3%. And our schedule strategy has only a 0.8% NoC bandwidth utilization rate gap.

C. Router Structure Design

To support the proposed workload balance arbiter and the behavior table based NoC routing algorithm, we design a corresponding router structure as shown in Figure 7. The router is performed in Store And Forward (SAF) mode [24], and all transfer data should be stored in the data buffer first. Then, the $Fetcher$ gets data from the data buffer based on the operations in the behavior table. The $Fetcher$ produces three outputs: the flag for data validity, the data priority, and the data packet. Afterward, the $Comparator$ compares the data priority with input priorities. To ensure the data validity, we set the transmission enable signal as the AND output of the flag and the $Comparator$ output. Finally, the $Data\ Transfer\ Unit$ distributes data to target ports for data transmission.

V. Experimental Results

A. Experiment Setup

We select AlexNet [25], VGG-8 [5], ResNet-18 [6], and VGG-16 on the CIFAR [9] and ImageNet [1] datasets as target DNNs. Networks on the CIFAR-10 and CIFAR-100 datasets only differ in the output dimensions of the last layer. Therefore the impact of differences on communication can be ignored, and we perform experiments only on the CIFAR-10 datasets. In the mapping process, besides our strategy, we select mapping methods proposed by MNSIM 2.0 [10] and Krishnan *et al.* [8] as the baseline mapping strategies. Besides, we adopt the mapping method in Liu *et al.* [15] as the control group of heuristic mapping algorithms. Moreover, in the schedule process, we utilize the XY routing algorithm [22] as the baseline schedule strategy. As for the hardware, we follow Multi-Precision [7] to construct PEs with four 256×256 crossbars and assign default values to other hardware hyperparameters. The target PIM architecture contains a 16×16 PE array in 2D-Mesh topology, with the typical NoC bandwidth in MNSIM 2.0 [10], i.e., 8 Gb/s. All experimental results are performed by the MNSIM 2.0 [10] in Intel® Xeon® E-5 2630 processors.

B. Comparison of Mapping and Schedule Strategies

Figure 8 shows the latency of AlexNet, VGG-8, ResNet-18, and VGG-16 on the CIFAR and ImageNet datasets under different mapping and schedule strategies. Compared with the baseline mapping methods proposed by MNSIM 2.0 and Krishnan *et al.*, the proposed mapping strategy reduces the latency by around 76% on average. And compared with the typical heuristic mapping method [15], our strategy still decreases the average latency by ∼56%. Furthermore, the proposed schedule strategy reduces the latency by around 60% on average over the XY [22] schedule method. Combining the proposed mapping and schedule strategies, we can always acquire the lowest latency, which is around 78% on average lower than the baseline strategies proposed by MNSIM 2.0 and Krishnan *et al.* Moreover, we evaluate our strategies' gap to the latency without communication conflicts. Experimental results in Figure 8 show the average gap is only 12%. In addition, the average latency gap on the ImageNet dataset is 6%, which means that our strategies can alleviate communication conflicts in complicated scenarios effectively.

Throughput comparison results are shown in Figure 9. Compared with the baseline mapping methods proposed by MNSIM 2.0 and Krishnan *et al.*, the proposed mapping strategy improves the throughput by around 3.29× on average. Besides, our mapping strategy enhances the average throughput by 2.57× over the typical heuristic mapping strategy [15]. Moreover, the proposed schedule strategy improves throughput to the XY schedule method by around 1.78× on average. Our strategies always reach the highest throughput, around 3.33× on average higher than the baseline strategies. And compared with throughput without communication conflicts, the average gap of our strategies is only 1.26%.

As shown in Figure 10, we compare the throughput of VGG-8 on the ImageNet dataset under different NoC bandwidth and different strategies. Our strategies significantly reduce the demand for NoC bandwidth and only need around 20% NoC bandwidth of baseline mapping and schedule methods [10] [15] to approach the throughput upper bound (6 Gb/s vs. 30 Gb/s). Even when the throughput reaches the upper bound (30 Gb/s), the proposed strategies can reduce the average communication energy consumption by 68.6% on variant DNNs and datasets.

C. Overhead of Mapping and Schedule Strategies

Table II shows the average overhead in search time consumption, hardware area, and energy cost of our mapping and schedule strategies on variant networks and datasets. In the mapping process, introducing the GNN based estimator brings about 11% search time consumption overhead. The reason for such low overhead is that the input communication conflict

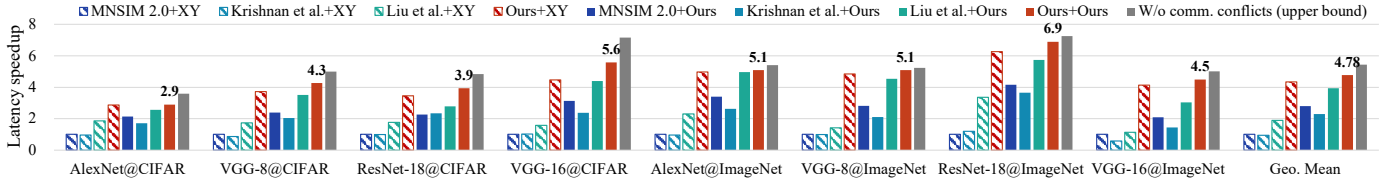

Fig. 8. Latency comparison under different mapping and schedule strategies on variant networks and datasets.

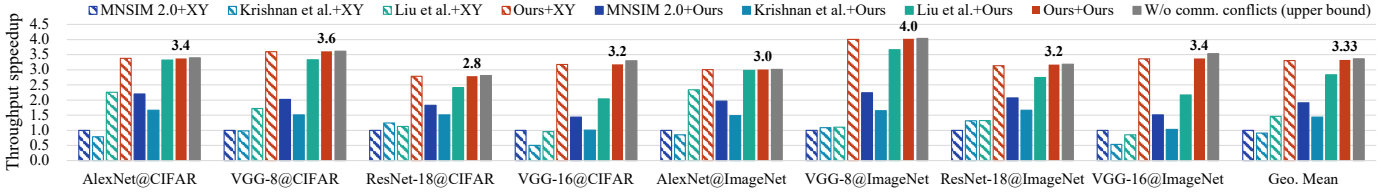

Fig. 9. Throughput comparison under different mapping and schedule strategies on variant networks and datasets.

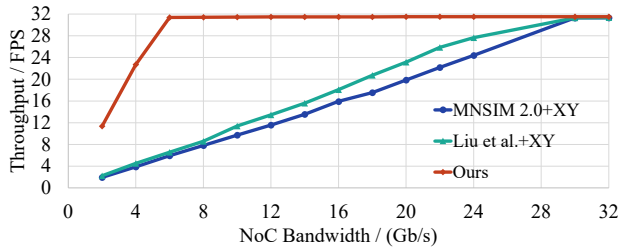

Fig. 10. Throughput of VGG-8@ImageNet under different NoC bandwidth.

TABLE II
SEARCH TIME, HARDWARE AREA, AND ENERGY COST OVERHEAD OF OUR MAPPING AND SCHEDULE STRATEGIES

Module	Search time/s	Area/mm^2	Energy/mJ
GNN estimator+behavior table (Proportion in the total)	16.67 **(11%)**	2.8 **(0.24%)**	0.16 **(0.16%)**
Total of ours	151.4	1131.7	100.96

graph has only several dozen vertices, and the GNN based estimator needs only 1.67 ms to perform single inference. And in the schedule process, our schedule strategy plugs the behavior table into routers, with an additional area and energy overhead of 0.24% and 0.16%, respectively.

VI. Conclusions

Communication conflicts become the critical bottleneck in NoC-based PIM architectures, harming the latency and throughput. We also find that only exactly modeling the communication conflicts can reach the hardware performance upper bound. This paper proposes communication conflict optimized mapping and schedule strategies to model and minimize communication conflicts. Our strategies leverage the GNN based estimator and the behavior table based routing algorithm to improve hardware performance. Compared with existing work, our strategies reduce 78% latency and improve the throughput by 3.33× on average with negligible deployment and hardware overhead. Besides, the proposed strategies reduce the average gap to the ideal cases without communication conflicts from 80.7% and 70% to 12% and 1.26% for latency and throughput, respectively.

VII. Acknowledgement

This work was supported by National Natural Science Foundation of China (No. 61832007, 62104128, U19B2019, U21B2031), Tsinghua University Initiative Scientific Research Program, Beijing National Research Center for Information Science and Technology (BNRist), and Tsinghua EE Xilinx AI Research Fund.

References

[1] J. Deng *et al.*, "Imagenet: A large-scale hierarchical image database," in *CVPR*, 2009.
[2] A. Vaswani *et al.*, "Attention is all you need," *NeurIPS*, 2017.
[3] P. Chi *et al.*, "Prime: A novel processing-in-memory architecture for neural network computation in reram-based main memory," *ACM SIGARCH Computer Architecture News*, 2016.
[4] M. Cheng *et al.*, "Time: A training-in-memory architecture for memristor-based deep neural networks," in *DAC*, 2017.
[5] K. Simonyan *et al.*, "Very deep convolutional networks for large-scale image recognition," *arXiv*, 2014.
[6] K. He *et al.*, "Deep residual learning for image recognition," in *CVPR*, 2016.
[7] Z. Zhu *et al.*, "A configurable multi-precision cnn computing framework based on single bit rram," in *DAC*, 2019.
[8] G. Krishnan *et al.*, "Impact of on-chip interconnect on in-memory acceleration of deep neural networks," *JETC*, 2021.
[9] A. Krizhevsky *et al.*, "Learning multiple layers of features from tiny images," 2009.
[10] Z. Zhu *et al.*, "Mnsim 2.0: A behavior-level modeling tool for memristor-based neuromorphic computing systems," in *GVLSI*, 2020.
[11] T. Maqsood *et al.*, "Congestion-aware core mapping for network-on-chip based systems using betweenness centrality," *FUTURE GENER COMP SY*, 2018.
[12] G. Krishnan *et al.*, "Interconnect-aware area and energy optimization for in-memory acceleration of dnns," *IEEE D&T*, 2020.
[13] S. K. Mandal *et al.*, "A latency-optimized reconfigurable noc for in-memory acceleration of dnns," *IEEE J EM SEL TOP C*, 2020.
[14] K. Zhou *et al.*, "Domino: A tailored network-on-chip architecture to enable highly localized inter-and intra-memory dnn computing," *arXiv*, 2021.
[15] J. Liu *et al.*, "A novel scheme to map convolutional networks to network-on-chip with computing-in-memory nodes," in *ISOCC*, 2020.
[16] Y. Ji *et al.*, "Fpsa: A full system stack solution for reconfigurable reram-based nn accelerator architecture," in *ASPLOS*, 2019.
[17] K. Deb *et al.*, "A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: Nsga-ii," in *PPSN*, 2000.
[18] T. N. Kipf *et al.*, "Semi-supervised classification with graph convolutional networks," *arXiv*, 2016.
[19] D. U. Becker, *Efficient microarchitecture for network-on-chip routers*, 2012.
[20] Y. Liu *et al.*, "A dynamic adaptive arbiter for network-on-chip," *Informacije MIDEM*, 2013.
[21] J. A. Bondy *et al.*, *Graph theory with applications*. Macmillan London, 1976.
[22] W. Zhang *et al.*, "Comparison research between xy and odd-even routing algorithm of a 2-dimension 3x3 mesh topology network-on-chip," in *WRI GCIS*, 2009.
[23] K. Guo *et al.*, "Rram based buffer design for energy efficient cnn accelerator," in *ISVLSI*, 2018.
[24] K. Jetly, "Experimental comparison of store-and-forward and wormhole noc routers for fpga's," 2013.
[25] A. Krizhevsky *et al.*, "Imagenet classification with deep convolutional neural networks," *Communications of the ACM*, 2017.