

# Communication-Aware Hybrid Parallelism Mapping for Low-Cost MCM-based DNN Accelerators

Jicheon Kim, Chunmyung Park, Xuan Truong Nguyen\* and Hyuk-Jae Lee\*

Department of Electrical and Computer Engineering, Seoul National University

Email: {jckim, lukpcm, truonngx, hyuk\_jae\_lee}@capp.snu.ac.kr

**Abstract**—The growing scale of deep neural networks has surpassed the capacity of single-chip accelerators, particularly pin cost-sensitive edge devices. Multi-Chip-Module (MCM) architectures enable scalability but rely on bandwidth-limited chip-to-chip (C2C) interfaces, causing substantial inter-chip communication overhead. Among model-parallel strategies, tensor parallelism (TP) offers high concurrency at the cost of communication overhead, while pipeline parallelism (PP) reduces it at the cost of lower compute utilization inherent to pipeline execution. This work presents *Stitch*, a two-phase rebalancing framework for hybrid model-parallel mapping in low-cost MCM-based CNN accelerators. Phase I mitigates TP’s C2C-induced communication overhead by jointly optimizing partitioning and datapath through a layer-wise C2C-DRAM selection solved via dynamic programming. Since TP alone cannot fully minimize communication, Phase II extends the design space by combining TP and PP at the package level. Guided by simulated annealing, *Stitch* selects layer groups, tunes pipeline stages, and balances communication-*utilization trade-offs*. Evaluation on a cycle-accurate simulator shows that *Stitch* reduces the energy-delay product by up to 42.8% compared to prior TP-based methods, demonstrating its effectiveness under practical C2C bandwidth constraints.

## I. INTRODUCTION

The rapid growth of deep neural networks (DNNs) has increased computational and memory requirements beyond what conventional single-chip System-on-Chip (SoC) accelerators can efficiently support, especially in cost-sensitive edge devices. Scaling single-chip solutions to advanced process nodes faces challenges such as reduced yield, higher manufacturing cost, and power inefficiency, which limit their practicality. Multi-Chip-Module (MCM) architectures, which integrate multiple smaller dies into a single package, have therefore emerged as a cost-effective alternative for scalability [1]–[3]. However, when these systems rely on low-cost chip-to-chip (C2C) interfaces, both limited bandwidth and the relatively high per-bit energy of inter-chip transfers can become major bottlenecks [4]. Addressing this constraint is essential for practical MCM-based DNN accelerators in edge environments.

Extensive research efforts have been made to improve workload partitioning and mapping in MCM-based accelerators. Model parallelism, in the form of tensor parallelism (TP), distributes intra-layer computations across chiplets to maximize concurrency [5]–[8]. In addition, hierarchical strategies that combine TP at the chiplet level with pipeline parallelism (PP) at the package level have been explored in large-scale server systems, primarily aiming to increase inference throughput [9].

These works demonstrated the potential of model-parallel mappings to scale performance well beyond the single-chip limit.

Despite extensive research on MCM-based accelerators, prior studies have largely focused on TP as the primary mapping strategy [5]–[8]. TP distributes intra-layer computations across chiplets to maximize concurrency, but it inherently incurs substantial communication overhead because multiple chiplets must access the same input data in parallel, which increases the total off-chip volume due to redundancy. Existing works typically assume that this data sharing is performed only through C2C links [5]–[8], without considering shared off-chip DRAM as an alternative datapath with distinct energy-latency characteristics. In bandwidth-constrained MCMs, such an assumption magnifies the cost of redundant transfers, leading to significant inefficiency in TP-centric mappings. This highlights the need to also consider DRAM as a complementary datapath in order to better balance bandwidth and energy trade-offs.

For hybrid approaches combining TP and PP, prior studies primarily target server-grade systems in which each chiplet integrates many cores [9]. Under such conditions, most communication can be naturally confined within chiplets via TP, while PP across chiplets is used to limit inter-die traffic and improve throughput, often supported by multi-batch inference to exploit data parallelism [10]. This separation is effective in server environments. In contrast, low-cost edge MCMs target latency-sensitive single-batch inference using small chiplets connected by bandwidth- and energy-constrained C2C links, where chiplet-level TP provides limited benefit. As a result, flexibly applying TP and PP at the package level becomes essential to balance utilization and communication overhead under strict system constraints.

To address this gap, we introduce *Stitch*, a communication-aware mapping framework that reduces communication overhead and improves energy-delay efficiency for single-batch inference on low-cost MCM accelerators. The contributions are summarized as follows:

- **Communication-Aware TP Rebalancing:** A dynamic programming-based method that determines per-layer TP partitioning and datapath selection (C2C vs. DRAM), providing optimal configurations under the cost model.
- **Hybrid TP/PP Rebalancing:** A simulated annealing-based exploration of hybrid mapping at the package level, adaptively applying TP or PP per layer while tuning pipeline stage granularity to balance utilization and communication overhead.

\* Corresponding authors.

- **System-Level Evaluation:** A cycle-accurate simulator of a low-cost MCM accelerator demonstrates that Stitch reduces energy–delay product (EDP) by up to 42.8% compared to prior TP-based mappings, validating its effectiveness for bandwidth-limited edge environments.

## II. BACKGROUND

### A. DNN Workload Mapping for MCM-based Accelerators

Efficient partitioning and mapping of DNN workloads are essential for achieving high performance in MCM-based accelerators. In this context, the computation workload—often described as a multi-dimensional index space  $J$  (e.g., nested loops in a CNN layer [11])—must be partitioned and mapped across available processors (chiplets) and time steps.

Formally, the mapping problem can be represented as  $T = \begin{bmatrix} \Pi \\ S \end{bmatrix}$ , an affine transform where  $\Pi : J \rightarrow \mathbb{Z}$  maps operations to time, and  $S : J \rightarrow \{0, \dots, M-1\}$  maps operations to chiplets;  $T$  must be injective and respect data dependencies [12], [13]. In single-chip accelerators, communication between cores is handled through on-chip interconnects. In contrast, MCM-based accelerators rely on C2C interconnects and a shared off-chip DRAM via the IO die, as depicted in Fig. 1. These off-chip transfers introduce significant data movement overhead, making communication-aware mapping crucial. Thus, the partitioning process involves (i) dividing the overall index space  $J$  into subsets  $\{P_1, P_2, \dots, P_M\}$ , each assigned to a chiplet, and (ii) determining a space–time mapping  $T(P_i)$  for each partition that minimizes a target cost such as  $Energy(T) \times Delay(T)$ .

Parallelism in DNNs is commonly categorized into data parallelism [10] and model parallelism [14]. Data parallelism replicates the model across devices and distributes training or inference samples, but its benefits are limited in single-batch inference, which is the target scenario in low-cost MCMs. In such cases, model parallelism—distributing computations of the same input across devices—is more appropriate. Within model parallelism, two representative strategies highlight different dimensions of partitioning. TP emphasizes spatial decomposition, dividing intra-layer operations across multiple processors to maximize concurrency. PP emphasizes temporal decomposition, assigning different layers (or groups of layers) to different processors and executing them in sequence. These two represent the spatial and temporal extremes of model-parallel mapping. In practice, most mapping schemes fall somewhere between these extremes, combining aspects of TP and PP, which motivates the need for frameworks that can flexibly explore this space in the MCM context.

### B. Related Works

TP has been the dominant approach for scaling DNNs, where intra-layer operations are distributed across multiple devices [10], [15]–[19]. While TP improves throughput, it inherently incurs substantial inter-chip communication.

Recent MCM-based accelerators extended TP concepts to chiplet architectures [5]–[8]. Simba [5] and Zimmer et al. [6] adopt weight-centric mapping, generating heavy C2C traffic from partial-sum exchanges. NN-Baton [7] localizes

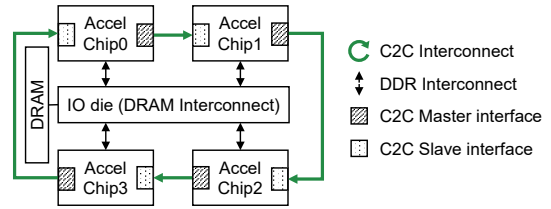


Fig. 1. A low-cost MCM-based DNN accelerator employing a half-duplex ring C2C interconnect [4].

partial-sum accumulation with output-centric partitioning, and INDM [8] further considers inter-layer transitions. However, these works assume high-speed interconnects and omit C2C–DRAM trade-offs critical in bandwidth-limited systems.

PP localizes each layer to a single chip, avoiding intra-layer exchanges but suffering from fill–drain latency and poor utilization. In multi-batch inference or training, micro-batching can improve utilization by overlapping different batches across pipeline stages [20]–[23]. However, in latency-sensitive single-batch inference, such overlap is impractical, leaving pipeline bubbles. To mitigate this, Stream [24] proposed fine-grained partitions such as single-row splits, which increase utilization by activating more stages concurrently. However, finer splits also create more inter-chip boundaries, and each boundary requires halo exchanges of feature maps, which can increase communication overhead under bandwidth limits.

Hybrid schemes combining TP and PP have also been explored. Tangram [25] integrates tile-level PP with intra-group TP, while Alpa [26] and [27] extend hierarchical model parallelism to GPUs. Gemini [9] applies package-level PP with chiplet-level TP for MCMs. These methods tend to adopt TP/PP hierarchies shaped by large-scale targets, without adaptive TP/PP selection or datapath rebalancing, which may reduce effectiveness under low-cost, bandwidth-constrained conditions.

These limitations highlight the need for a framework tailored to low-cost MCM accelerators that jointly optimizes TP/PP mapping and off-chip datapath selection.

## III. SYSTEM MODELING

### A. Parallelism Strategies in MCM

Although parallelism can be formulated more generally in the time–space domain, practical MCM accelerators are most often described by two representative extremes—TP and PP. These two strategies are emphasized here as they capture the key trade-off between utilization and communication overhead, the central challenge in bandwidth-constrained MCMs.

In TP, computations within a single layer are partitioned across multiple chips to maximize concurrency. Package-level partitioning determines how the workload is divided, while chiplet-level mapping optimizes the local dataflow. Fig. 2(a) illustrates the loop structure of a convolution layer under this decomposition. Two common partitioning strategies are planar partitioning (Fig. 2(b)), which requires weight sharing (WS) across chips, and channel partitioning (Fig. 2(c)), which requires feature-map sharing (FS). These redundant transfers (WS or FS) are the primary source of inter-chip communication

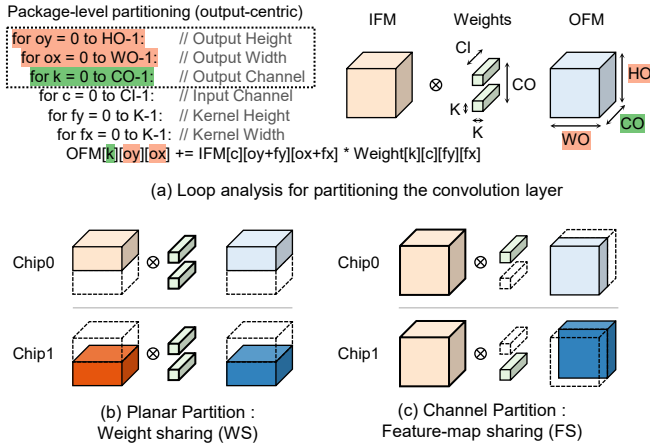


Fig. 2. Package-level tensor parallelism: based on planar and channel partitioning.

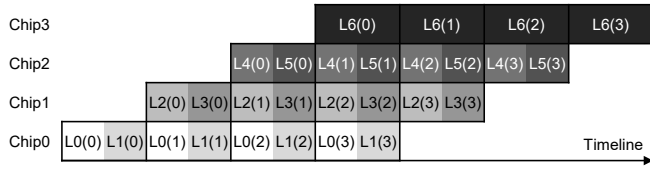


Fig. 3. Package-level pipeline parallelism: based on planar partitioning across four pipeline stages in a four-chip configuration, where each layer (L0–L6) is split into four partitions, denoted as L0(0)–L0(3), ..., L6(0)–L6(3).

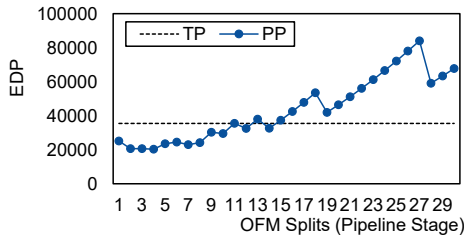


Fig. 4. EDP comparison between TP with planar partitioning and PP with varying OFM split sizes for ResNet-50 (Layers 5–10), evaluated on a target C2C bandwidth-constrained 4-chip MCM-based system configured in Table II and Table III.

overhead in TP, as formally quantified in Section III-B. To avoid even higher costs, input-channel partitioning (i.e., weight-centric mapping) [5], [6] is excluded in this work, and TP is restricted to output-centric partitioning only.

In contrast, PP localizes the entire computation of each layer to a single chip, eliminating redundant intra-layer data sharing and reducing inter-chip communication to unicast transfers at pipeline boundaries. However, it suffers from fill–drain latency and load imbalance, which are particularly severe under single-batch inference since inter-layer dependencies force each layer to complete before the next can start, leaving other chips idle. Channel partitioning in PP is especially problematic for this reason: unlike planar partitioning, which can divide a layer into row splits to form pipeline stages, channel partitioning cannot overlap execution across layers and therefore fails to improve utilization. For these reasons, our framework restricts

TABLE I  
INTER-CHIP COMMUNICATION PATTERNS IN MCM MAPPING

Partition switch	Intra-layer (WS)	Inter-layer (FS)
Planar → Planar	AG: $(1-1/M) W_i $	Unicast (halo): $r_{i+1} \cdot WO_i \cdot CO_i \cdot b_a$
Planar → Channel	AG: $(1-1/M) W_i $	AG: $(1-1/M) \cdot  FM_i $
Channel → Planar	–	Scatter: $((M-1)/M^2) \cdot  FM_i $
Channel → Channel	–	AG: $(1-1/M) \cdot  FM_i $

All communication volumes are expressed as per-chip amounts.  
 $|W_i| = CO_i \cdot CI_i \cdot K \cdot K \cdot b_w$ ;  $|FM_i| = HO_i \cdot WO_i \cdot CO_i \cdot b_a$ ;  $r_{i+1}$ : effective halo rows (= halo rows  $\times$  #adjacent partitions; 1 for edge, 2 for internal);  $WO_i$ : output width of  $FM_i$ ;  $CO_i$ : output channels;  $M$ : #chips;  $b_a$ : activation bits;  $b_w$ : weight bits.

PP to planar partitioning. To illustrate, Fig. 3 shows a case where the output feature-map (OFM) is divided into four row partitions across four chips. Here, Chip0 processes the first-stage chunks  $L0(0)$  and  $L1(0)$ , then forwards the resulting activations (including halo regions) to Chip1, which continues with  $L2(0)$  and  $L3(0)$  for the same partition index, and so on. In this way, each partition index flows through a sequence of chips as a pipeline stage.

However, the efficiency of PP strongly depends on the chosen partition granularity. Prior work such as [24], originally targeting single-chip multi-core systems, tried to mitigate utilization loss by enforcing extremely fine partitions (e.g., single-row). While finer splits activate more pipeline stages concurrently and reduce idle bubbles, they also increase the number of inter-chip boundaries, each of which requires halo exchanges. As shown in Fig. 4, this trade-off means that partition granularity must be carefully tuned in MCM accelerators to achieve a balance between higher utilization and rising communication overhead.

### B. Communication Cost Model

Following prior studies [8], [15], [26], we model inter-chip communication to capture the overhead of different parallelism strategies. Table I summarizes three primitives: *All-gather* (AG), where each chip holds only a  $1/M$  fraction of a tensor and must gather the remaining parts (weights or feature maps) from peers; *Unicast*, a one-to-one halo exchange in planar partitioning where boundary rows are fetched from neighbors; and *Scatter*, which arises when the partitioning method changes (e.g., Channel → Planar), requiring each chip’s  $1/M$  share in a mismatched shape to be redistributed to all others.

For PP, intra-layer redundancy is eliminated since each layer is localized, but stage-to-stage transfers remain at pipeline boundaries (Fig. 3), and the resulting overhead depends on the chosen pipeline granularity. From these primitives, the total off-chip data volume for each strategy can be expressed as: TP–Planar incurs  $M|W_i| + |FM_i| + |H_i|$ , TP–Channel yields  $|W_i| + M|FM_i|$ , while PP–Planar requires only  $|W_i| + |FM_i| + |H_i|$ , where  $|H_i|$  denotes the total halo exchange volume. Thus, TP inevitably multiplies either weights or feature maps in proportion to  $M$ , whereas PP localizes data and avoids such redundancy at the cost of inter-stage transfers.

To evaluate partitioning strategies, we model the total off-chip communication cost, which includes inter-chip sharing

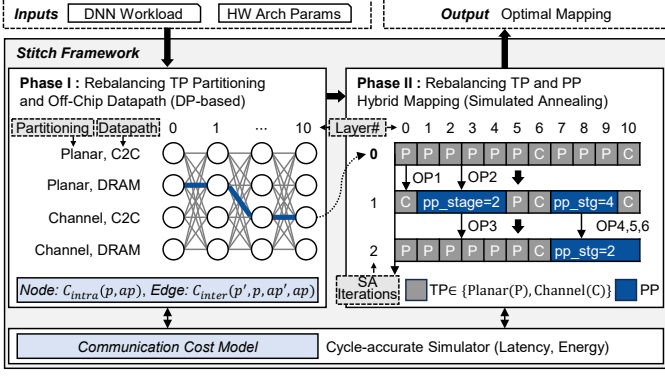


Fig. 5. Stitch framework: Phase I performs per-layer TP partitioning with C2C/DRAM datapath selection, while Phase II explores hybrid TP/PP mappings and pipeline granularity.

through both C2C and DRAM, as well as partition-dependent DRAM initial loading and spill/reload under limited L2 capacity (on-chip SRAM used as the last-level buffer). The cost is expressed as

$$C = (E_{C2C} + E_{DRAM}) \times (L_{C2C} + L_{DRAM}), \quad (1)$$

where  $V$  denotes the transfer volume and  $B$  the available bandwidth of a datapath. The latency of a transfer is  $L = V/B$ , and the total energy is  $E = e \cdot V$ , with  $e$  denoting the per-bit energy.

Each CNN layer  $i$  is associated with two mapping decisions: the partitioning method  $p_i \in \{\text{Planar}, \text{Channel}\}$  and the off-chip access path  $ap_i \in \{\text{C2C}, \text{DRAM}\}$ . Given these decisions, the communication cost for layer  $i$  is decomposed into intra-layer and inter-layer terms:

- $C_{\text{intra}}(p_i, ap_i)$ : intra-layer cost of layer  $i$ , determined solely by its own partitioning and access path. It includes all off-chip transfers intrinsic to the layer, such as redundant weight sharing (Table I, WS), first-layer input distribution, and additional DRAM traffic from L2 spill.
- $C_{\text{inter}}(p_{i-1}, p_i, ap_{i-1}, ap_i)$ : inter-layer transition cost, determined by partition switching between consecutive layers. It corresponds to feature-map sharing (Table I, FS) operations such as halo exchanges, all-gather, or scatter when outputs must be redistributed across chips.

#### IV. PARTITIONING AND MAPPING FRAMEWORK

##### A. Stitch Overview

Fig. 5 illustrates Stitch, a two-phase framework for partitioning and mapping CNN workloads on MCM accelerators under bandwidth-constrained C2C links. The framework reduces communication overhead by rebalancing data movement across C2C and DRAM, and by flexibly combining TP and PP.

**Phase I (TP Rebalancing).** Dynamic programming (DP) selects per-layer TP partitioning together with the off-chip datapath (C2C or DRAM).

**Phase II (Hybrid TP/PP Rebalancing).** Building on the TP configuration, simulated annealing (SA) explores hybrid mappings by choosing TP or PP per layer, grouping layers into PP stages, and tuning pipeline granularity.

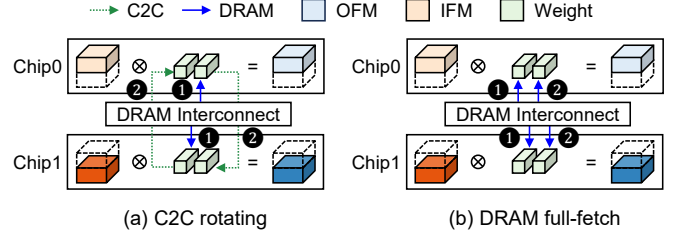


Fig. 6. Redundant weight sharing under planar TP partitioning (two chips, two weight blocks). (a) **C2C rotating**: Each chip fetches one weight block from DRAM and forwards it to the peer over C2C. (b) **DRAM full-fetch**: Each chip fetches both weight blocks from DRAM.

Both phases rely on a unified communication cost model and cycle-accurate simulation to ensure mapping decisions respect bandwidth and latency constraints.

##### B. TP Partitioning and Datapath Rebalancing

In Phase I, Stitch rebalances data sharing between C2C and DRAM datapaths while selecting per-layer TP partitioning. Unlike prior TP-based mapping methods that assume C2C as the only path, Stitch explicitly evaluates both C2C and DRAM, enabling more efficient trade-offs under bandwidth constraints.

As illustrated in Fig. 1, data sharing between chiplets can occur through directly connected C2C interfaces or via shared DRAM. For example, when the OFM is partitioned along the height dimension and distributed across two chips, redundant weights must be available to both. In the case of two chips and two weight blocks (Fig. 6), the *C2C rotating* scheme requires each chip to fetch one block from DRAM ( $W$  per chip) and forward it to its peer over C2C ( $W$  per chip), while the *DRAM full-fetch* scheme requires each chip to fetch both blocks directly from DRAM ( $2W$  per chip) without any C2C transfers. Using the energy and bandwidth parameters in Tables II–III ( $e_{\text{DRAM}}=8.75$  pJ/bit,  $e_{\text{C2C}}=0.83$  pJ/bit,  $B_{\text{DRAM}}=37.5$  Gb/s,  $B_{\text{C2C}}=2$  Gb/s), the corresponding costs from (1) are

$$C_{\text{rot}} \propto (e_{\text{C2C}} \cdot W + e_{\text{DRAM}} \cdot W) \times \left( \frac{W}{B_{\text{C2C}}} + \frac{W}{B_{\text{DRAM}}} \right),$$

$$C_{\text{full}} \propto (e_{\text{DRAM}} \cdot 2W) \times \left( \frac{2W}{B_{\text{DRAM}}} \right).$$

Under these settings, the full-fetch option achieves an illustrative  $\sim 82\%$  lower cost than rotating.

The optimization is formulated as a DP. Because the total cost decomposes into additive intra-layer and inter-layer terms that depend only on the current and immediately preceding layer, the problem satisfies the Markov property. Since computation cost is invariant under TP regardless of partitioning, only communication cost is considered, and the recurrence is expressed as

$$DP[i][p][ap] = \min_{p', ap'} \left[ DP[i-1][p'][ap'] + C_{\text{intra}}(p, ap) + C_{\text{inter}}(p', p, ap', ap) \right]. \quad (2)$$

Here, the state  $(i, p, ap)$  is updated from all possible predecessors  $(i-1, p', ap')$ , and the optimal sequence  $\{p_i, ap_i\}$  under the defined cost model is obtained by backtracking from the final state with minimum cost.

### C. Hybrid TP/PP Rebalancing

In Phase II, Stitch rebalances the trade-off between TP and PP at the package level. While TP offers high utilization at the cost of redundant communication, PP reduces inter-chip traffic but suffers from fill–drain overhead and load imbalance. Phase I provides an optimal TP-only mapping under the defined cost model, and additional gains can be achieved by incorporating PP at the package level. To explore this hybrid space, Stitch employs SA guided by a detailed cost evaluation. SA is preferred to refine the Phase I initialization via local perturbations that align with the problem’s sequential structure. Population-based metaheuristics (e.g., genetic algorithms) require complex constraint handling to avoid invalid pipeline configurations and, under such constraints, effectively degenerate to SA-like local search with additional overhead. We define six neighborhood operators to restructure TP/PP mappings:

- **OP1 (flip\_tp)**: Toggle the partitioning method of one TP layer (Planar↔Channel).
- **OP2 (convert\_tp\_to\_pp)**: Convert a run of TP layers into one PP group with a sampled stage count.
- **OP3 (convert\_pp\_to\_tp)**: Dismantle a PP group into TP layers, restoring their original partitioning.
- **OP4 (modify\_pp\_stage)**: Perturb the stage count of a PP group (e.g.,  $\pm 1$ ) within legal bounds.
- **OP5 (extend\_pp\_group)**: Absorb one adjacent TP layer into a PP group and resample its stage allocation.
- **OP6 (shrink\_pp\_group)**: Remove a terminal PP layer, revert it to TP, and adjust stage count if required.

These operators allow SA to flexibly convert between TP and PP, merge or split PP groups, and adjust pipeline granularity. In addition, PP groups are subject to the L2 memory capacity of each chiplet. During pipeline execution, weights and intermediate feature maps are accumulated in the buffer; if their combined size exceeds the available L2 capacity, Stitch models the additional DRAM traffic required to spill and reload data. Specifically, intermediate OFMs that cannot be retained on-chip are written back to DRAM and fetched again in later stages, incurring both latency and energy penalties. This constraint highlights a general insight: pipeline depth cannot be increased arbitrarily. While finer partitioning may reduce pipeline bubbles, excessively deep PP groups quickly exceed L2 capacity and trigger DRAM spill, offsetting potential benefits.

## V. EXPERIMENTAL RESULTS

### A. Evaluation Settings

**Methodology.** We implement the cycle-accurate simulator shown at the bottom of Fig. 5, which models the ring-based MCM accelerator in Fig. 1 [4], targeting a low-cost edge system. Unlike prior baselines that assume high-bandwidth 16 nm ground-referenced signaling (GRS) links on organic substrates [28], denoted as C2C-H (25 Gb/s, 1.17 pJ/bit), our simulator incorporates a low-cost parallel C2C interface in 16 nm derived from an advanced interface bus (AIB)-style implementation [29], denoted as C2C-L (2 Gb/s, 0.83 pJ/bit). The adopted energy model is summarized in Table II, where DRAM, SRAM, and MAC parameters follow prior 16

TABLE II  
ENERGY CONSUMPTION AT 16NM [7], [8], [29]

Operation	Energy (pJ/bit)
Inter-die communication	0.83
DRAM access	8.75
SRAM access	0.81
8bit MAC	0.024

TABLE III  
SYSTEM CONFIGURATION FOR SIMULATION

Parameter	Value
C2C bandwidth per pin (Gbps)	2
DRAM bandwidth (Gbps/chiplet)	37.5
The number of chiplets	4
The number of MACs per chiplet	1024
Clock frequency (GHz)	1
L2 size (KB)	1024

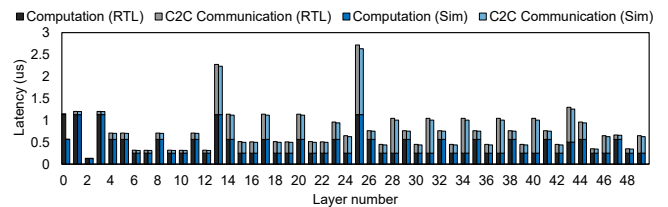


Fig. 7. Layer-wise latency comparison of ResNet-50 between the implemented MCM simulator and the RTL design in [4].

nm baselines [7], [8], while inter-die communication energy comes from the AIB-style implementation [29]. The overall system configuration is listed in Table III, with per-link C2C bandwidth taken from the same AIB-style design, and the remaining parameters aligned with baseline studies to ensure fair comparison. In addition, pipeline depth is constrained by the L2 capacity as discussed in Section IV-C, where excessive stages may trigger DRAM spill and offset potential benefits.

**Validation.** To ensure accuracy of our evaluation framework, we compare the cycle-accurate simulator against RTL implementation [4] on ResNet-50 [30]. As shown in Fig. 7, the average error is 2.41% for computation, 5.06% for C2C communication, and 3.53% overall. The remaining discrepancy is mainly due to on-chip/off-chip protocol conversion and burst-transaction overhead present only in the RTL C2C interface.

**Baseline.** For comparison, we adopt NN-Baton [7] and INDM [8] as baselines. NN-Baton heuristically selects per-layer TP partitions based only on intra-layer communication cost. INDM applies DP that considers both intra-layer and inter-layer communication costs. The proposed methods are denoted as Stitch (Phase I only), which rebalances TP with DRAM/C2C path selection, and Stitch (Phase I&II), which further combines TP rebalancing with hybrid TP/PP mapping. Unlike mesh-topology approaches for large-scale systems (e.g., Gemini [9]), such baselines are excluded since our target is single-batch inference on low-cost ring-based MCMs. Package-level PP is instead explored within Stitch (Phase II).

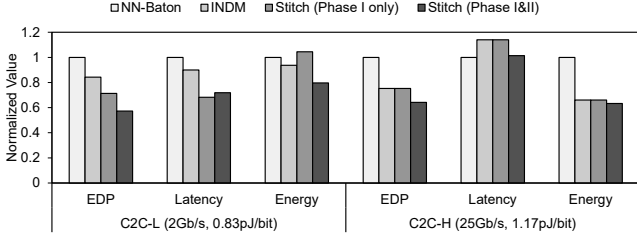


Fig. 8. End-to-end ResNet-50 inference results comparing normalized EDP, latency, and energy across two baselines (NN-Baton, INDM) and the proposed methods (Stitch Phase I only and Stitch Phase I&II).

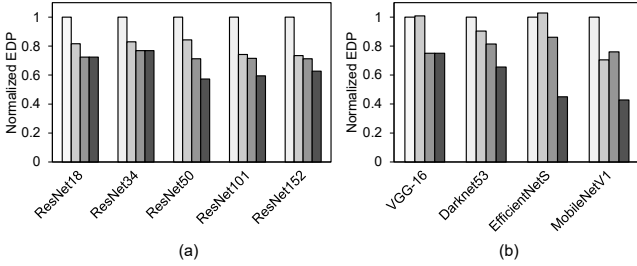


Fig. 9. EDP comparison under different mapping strategies: (a) ResNet models of varying scale, and (b) diverse CNN models.

### B. End-to-End Performance

Fig. 8 compares proposed methods with baselines on ResNet-50. With C2C-L, Phase I achieves a 28.7% EDP reduction over NN-Baton and 15.4% over INDM by rebalancing datapaths between C2C and DRAM. Although DRAM consumes higher per-bit energy than C2C, its larger bandwidth reduces latency, and rebalancing these trade-offs leads to overall improvement. Adding Phase II yields a further 19.7% reduction relative to Phase I, resulting in 42.8% and 32.1% reductions compared to NN-Baton and INDM, respectively. This improvement comes from PP, which reduces redundant communication (lowering energy) at the cost of some utilization (increasing latency), but overall achieves a better energy–latency balance. Under C2C-H, Phase I shows no gain over INDM and Phase II achieves only 14.7%, which was 19.7% under C2C-L, indicating that while high-speed links already favor existing approaches, datapath rebalancing is particularly effective in bandwidth-constrained C2C, where prior methods are suboptimal.

Fig. 9 provides further insight across different network architectures. In Fig. 9(a), ResNet scaling illustrates how network depth affects hybrid mapping. For shallow networks such as ResNet-18 and ResNet-34, PP provides no additional gain, and Phase II does not select any PP grouping. As depth increases (ResNet-101 and ResNet-152), later layers become dominated by weight size relative to activations, leading to more channel partitions. In these cases, some planar partitions are grouped into PP, but the fraction of such layers compared to the total decreases, resulting in smaller overall EDP improvements.

In Fig. 9(b), other CNN models show distinct behaviors. VGG-16 [31], with relatively few layers, likewise shows no gain from PP. DarkNet53 [32] behaves similarly to ResNet due to its depth and weight-dominant later layers. In contrast,

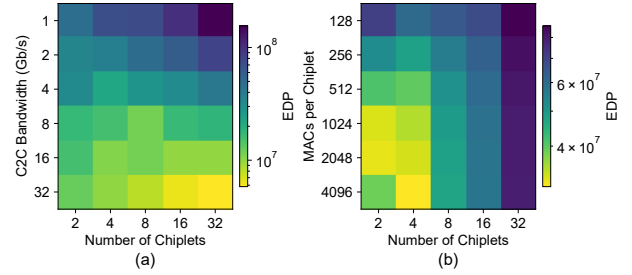


Fig. 10. EDP exploration under different system scaling factors: (a) chiplet count vs. C2C bandwidth, and (b) chiplet count vs. MACs per chiplet.

depthwise-separable architectures such as EfficientNetS [33] and MobileNetV1 [34] exhibit communication-dominant characteristics. For these models, datapath rebalancing alone yields limited effect, but PP substantially reduces communication overhead, producing the largest gains. These results highlight that the effectiveness of TP/PP hybrid mapping strongly depends on network depth and the communication characteristics determined by the relative size of weights and activations.

### C. Design Space Exploration

Fig. 10(a) shows that when C2C bandwidth is limited, reducing the number of chiplets yields lower EDP, whereas with higher bandwidth, scaling to more chiplets becomes advantageous. Fig. 10(b) illustrates that chiplet-level compute granularity can be tuned under area constraints to meet a target EDP at minimal cost. The framework enables rapid exploration, requiring on average about 57 s for 100,000 SA iterations on an Intel Xeon Silver 4214 processor (2.2 GHz) running Ubuntu 18.04, compared to several hours typically needed for RTL simulation.

## VI. CONCLUSION

This work addressed communication bottlenecks in low-cost MCM-based CNN accelerators with bandwidth-constrained C2C links. We presented Stitch, a two-phase framework: Phase I optimizes TP partitioning with C2C/DRAM datapath selection using dynamic programming, and Phase II integrates PP with adaptive pipeline granularity via simulated annealing. Unlike prior TP-centric methods, Stitch models DRAM as a complementary datapath and flexibly applies PP at the package level. Experiments on ResNet-50 and other CNNs show up to 42.8% EDP reduction under constrained settings, demonstrating the practicality of communication-aware partitioning with TP/PP integration for scalable edge inference.

### ACKNOWLEDGMENT

This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) under the Graduate School of Artificial Intelligence Semiconductor(IITP-2026-RS-2023-00256081) grant funded by the Korea government (MSIT) and in part by the IITP grant funded by MSIT (No.RS 2023-00228255, PIM-NPU Based Processing System Software Developments for Hyperscale Artificial Neural Network Processing).

## REFERENCES

- [1] G. H. Loh and R. Swaminathan, "The Next Era for Chiplet Innovation," in *2023 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, Apr. 2023, pp. 1–6.
- [2] H. Sharma, S. K. Mandal, J. R. Doppa, U. Ogras, and P. P. Pande, "Achieving Datacenter-scale Performance through Chiplet-based Many-core Architectures," in *2023 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, Apr. 2023, pp. 1–6.
- [3] H. Sharma, G. Narang, J. R. Doppa, U. Ogras, and P. P. Pande, "Dataflow-Aware PIM-Enabled Manycore Architecture for Deep Learning Workloads," in *2024 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, Mar. 2024, pp. 1–6.
- [4] J. Kim, C. Park, E. Hyun, X. T. Nguyen, and H.-J. Lee, "A Highly-Scalable Deep-Learning Accelerator With a Cost-Effective Chip-to-Chip Adapter and a C2C-Communication-Aware Scheduler," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 14, no. 3, pp. 455–468, Sep. 2024.
- [5] Y. S. Shao, J. Clemons, R. Venkatesan, B. Zimmer, M. Fojtik, N. Jiang, B. Keller, A. Klinefelter, N. Pinckney, P. Raina, S. G. Tell, Y. Zhang, W. J. Dally, J. Emer, C. T. Gray, B. Khailany, and S. W. Keckler, "Simba: Scaling Deep-Learning Inference with Multi-Chip-Module-Based Architecture," in *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, Oct. 2019, pp. 14–27.
- [6] B. Zimmer, R. Venkatesan, Y. S. Shao, J. Clemons, M. Fojtik, N. Jiang, B. Keller, A. Klinefelter, N. Pinckney, P. Raina, S. G. Tell, Y. Zhang, W. J. Dally, J. S. Emer, C. T. Gray, S. W. Keckler, and B. Khailany, "A 0.32–128 TOPS, Scalable Multi-Chip-Module-Based Deep Neural Network Inference Accelerator With Ground-Referenced Signaling in 16 nm," *IEEE Journal of Solid-State Circuits*, vol. 55, no. 4, pp. 920–932, Apr. 2020.
- [7] Z. Tan, H. Cai, R. Dong, and K. Ma, "NN-Baton: DNN Workload Orchestration and Chiplet Granularity Exploration for Multichip Accelerators," in *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*, Jun. 2021, pp. 1013–1026.
- [8] J. Zhang, X. Fan, Y. Ye, X. Wang, G. Xiong, X. Leng, N. Xu, Y. Lian, and G. He, "INDM: Chiplet-Based Interconnect Network and Dataflow Mapping for DNN Accelerators," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 43, no. 4, pp. 1107–1120, Apr. 2024.
- [9] J. Cai, Z. Wu, S. Peng, Y. Wei, Z. Tan, G. Shi, M. Gao, and K. Ma, "Gemini: Mapping and Architecture Co-exploration for Large-scale DNN Chiplet Accelerators," in *2024 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, Mar. 2024, pp. 156–171.
- [10] J. Dean, G. S. Corrado, R. Monga, K. Chen, M. Devin, Q. V. Le, M. Z. Mao, M. Ranzato, A. Senior, P. Tucker, K. Yang, and A. Y. Ng, "Large scale distributed deep networks," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2012, pp. 1223–1231.
- [11] S. Lee, B. Kim, Y. Choi, and H.-J. Lee, "HopScotch: A Holistic Approach to Data Layout-Aware Mapping on NPUs for High-Performance DNN Inference," *ACM Transactions on Architecture and Code Optimization*, vol. 22, no. 3, pp. 1–26, Sep. 2025.
- [12] D. I. Moldovan and J. A. B. Fortes, "Partitioning and mapping algorithms into fixed size systolic arrays," *IEEE Transactions on Computers*, vol. 35, no. 1, pp. 1–12, Jan. 1986.
- [13] H.-J. Lee and J. A. B. Fortes, "Generation of injective and reversible modular mappings," *IEEE Transactions on Parallel and Distributed Systems*, vol. 14, no. 1, pp. 1–12, Jan. 2003.
- [14] A. Krizhevsky, "One weird trick for parallelizing convolutional neural networks," 2014, [arXiv:1404.5997](https://arxiv.org/abs/1404.5997).
- [15] Z. Jia, S. Lin, C. R. Qi, and A. Aiken, "Exploring hidden dimensions in accelerating convolutional neural networks," in *Proceedings of the 35th International Conference on Machine Learning*, vol. 80, Jul. 2018, pp. 2274–2283.
- [16] L. Song, J. Mao, Y. Zhuo, X. Qian, H. Li, and Y. Chen, "HyPar: Towards hybrid parallelism for deep learning accelerator array," in *2019 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, 2019, pp. 56–68.
- [17] L. Song, F. Chen, Y. Zhuo, X. Qian, H. Li, and Y. Chen, "AccPar: Tensor Partitioning for Heterogeneous Deep Learning Accelerators," in *2020 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, Feb. 2020, pp. 342–355.
- [18] M. Shoeybi, M. Patwary, R. Puri, P. LeGresley, J. Casper, and B. Catanzaro, "Megatron-LM: Training Multi-Billion Parameter Language Models Using Model Parallelism," 2019, [arXiv:1909.08053](https://arxiv.org/abs/1909.08053).
- [19] Y. Xu, H. Lee, D. Chen, B. Hechtman, Y. Huang, R. Joshi, M. Krikun, D. Lepikhin, A. Ly, M. Maggioni, R. Pang, N. Shazeer, S. Wang, T. Wang, Y. Wu, and Z. Chen, "GSPMD: General and Scalable Parallelization for ML Computation Graphs," 2021, [arXiv:2105.04663](https://arxiv.org/abs/2105.04663).
- [20] Y. Huang, Y. Cheng, A. Bapna, O. Firat, M. X. Chen, D. Chen, H. Lee, J. Ngiam, Q. V. Le, Y. Wu, and Z. Chen, "GPipe: Efficient Training of Giant Neural Networks using Pipeline Parallelism," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2019, pp. 103–112.
- [21] D. Narayanan, A. Harlap, A. Phanishayee, V. Seshadri, N. R. Devanur, G. R. Ganger, P. B. Gibbons, and M. Zaharia, "PipeDream: Generalized pipeline parallelism for DNN training," in *Proceedings of the 27th ACM Symposium on Operating Systems Principles (SOSP)*, Oct. 2019, pp. 1–15.
- [22] S. Fan, Y. Rong, C. Meng, Z. Cao, S. Wang, Z. Zheng, C. Wu, G. Long, J. Yang, L. Xia, L. Diao, X. Liu, and W. Lin, "Dapple: a pipelined data parallel approach for training large models," in *Proceedings of the 26th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPoPP)*, 2021, p. 431–445.
- [23] J. Yim, J. Song, Y. Choi, J. Lee, J. Jung, H. Jang, and J. Lee, "PipeTte: Automatic Fine-Grained Large Language Model Training Configurator for Real-World Clusters," in *2024 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, Mar. 2024, pp. 1–6.
- [24] A. Symons, L. Mei, S. Colleman, P. Houshmand, S. Karl, and M. Verhelst, "Stream: Design Space Exploration of Layer-Fused DNNs on Heterogeneous Dataflow Accelerators," *IEEE Transactions on Computers*, vol. 74, no. 1, pp. 237–249, Jan. 2025.
- [25] M. Gao, X. Yang, J. Pu, M. Horowitz, and C. Kozyrakis, "TANGRAM: Optimized Coarse-Grained Dataflow for Scalable NN Accelerators," in *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, Apr. 2019, pp. 807–820.
- [26] L. Zheng, Z. Li, H. Zhang, Y. Zhuang, Z. Chen, Y. Huang, Y. Wang, Y. Xu, D. Zhuo, E. P. Xing, J. E. Gonzalez, and I. Stoica, "Alpa: Automating Inter- and Intra-Operator Parallelism for Distributed Deep Learning," in *16th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, Jul. 2022, pp. 559–578.
- [27] Y. Zhuang, L. Zheng, Z. Li, E. Xing, Q. Ho, J. Gonzalez, I. Stoica, H. Zhang, and H. Zhao, "On optimizing the communication of model parallelism," in *Proceedings of Machine Learning and Systems*, vol. 5, 2023, pp. 526–540.
- [28] J. W. Poulton, J. M. Wilson, W. J. Turner, B. Zimmer, X. Chen, S. S. Kudva, S. Song, S. G. Tell, N. Nedovic, W. Zhao, S. R. Sudhakaran, C. T. Gray, and W. J. Dally, "A 1.17-pJ/b, 25-Gb/s/pin Ground-Referenced Single-Ended Serial Link for Off- and On-Package Communication Using a Process- and Temperature-Adaptive Voltage Regulator," *IEEE Journal of Solid-State Circuits*, vol. 54, no. 1, pp. 43–54, Jan. 2019.
- [29] C. Liu, J. Botimer, and Z. Zhang, "A 256Gb/s/mm-shoreline AIB-Compatible 16nm FinFET CMOS Chiplet for 2.5D Integration with Stratix 10 FPGA on EMIB and Tiling on Silicon Interposer," in *2021 IEEE Custom Integrated Circuits Conference (CICC)*, Apr. 2021, pp. 1–2.
- [30] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016, pp. 770–778.
- [31] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," 2014, [arXiv:1409.1556](https://arxiv.org/abs/1409.1556).
- [32] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," 2018, [arXiv:1804.02767](https://arxiv.org/abs/1804.02767).
- [33] S. Gupta and B. Akin, "Accelerator-aware Neural Network Design using AutoML," 2020, [arXiv:2003.02838](https://arxiv.org/abs/2003.02838).
- [34] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," 2017, [arXiv:1704.04861](https://arxiv.org/abs/1704.04861).