Gemini: Mapping and Architecture Co-exploration for Large-scale DNN Chiplet Accelerators

Jingwei Cai[†], Zuotong Wu[‡§], Sen Peng[‡§], Yuchen Wei[†], Zhanhong Tan[†], Guiming Shi[†], Mingyu Gao[†¶*] and Kaisheng Ma[†*]
Tsinghua University[†], Xi'an Jiaotong University[‡], IIISCT[§], Shanghai AI Laboratory[¶]
Corresponding Author[*]
{caijw21,gaomy,kaisheng}@tsinghua.edu.cn

2024 IEEE International Symposium on High-Performance Computer Architecture (HPCA) | 979-8-3503-9313-2/24/$31.00 ©2024 IEEE | DOI: 10.1109/HPCA57654.2024.00022

***Abstract*—Chiplet technology enables the integration of an increasing number of transistors on a single accelerator with higher yield in the post-Moore era, addressing the immense computational demands arising from rapid AI advancements. However, it also introduces more expensive packaging costs and costly Die-to-Die (D2D) interfaces, which require more area, consume higher power, and offer lower bandwidth than on-chip interconnects. Maximizing the benefits and minimizing the drawbacks of chiplet technology is crucial for developing large-scale DNN chiplet accelerators, which poses challenges to both architecture and mapping. Despite its importance in the post-Moore era, methods to address these challenges remain scarce.**

To bridge the gap, we first propose a layer-centric encoding method to encode Layer-Pipeline (LP) spatial mapping for large-scale DNN inference accelerators and depict the optimization space of it. Based on it, we analyze the unexplored optimization opportunities within this space, which play a more crucial role in chiplet scenarios. Based on the encoding method and a highly configurable and universal hardware template, we propose an architecture and mapping co-exploration framework, Gemini, to explore the design and mapping space of large-scale DNN chiplet accelerators while taking monetary cost (MC), performance, and energy efficiency into account. Compared to the state-of-the-art (SOTA) Simba architecture with SOTA Tangram LP Mapping, Gemini's co-optimized architecture and mapping achieve, on average, 1.98× performance improvement and 1.41× energy efficiency improvement simultaneously across various DNNs and batch sizes, with only a 14.3% increase in monetary cost. Moreover, we leverage Gemini to uncover intriguing insights into the methods for utilizing chiplet technology in architecture design and mapping DNN workloads under chiplet scenarios. *The Gemini framework is open-sourced at https://github.com/SET-Scheduling-Project/GEMINI-HPCA2024.*

I. Introduction

As Deep Neural Networks (DNNs) tackle increasingly complex problems, their size and complexity grow rapidly, resulting in increased computing and storage demands [9], [10], [51], [56], [63]. While applying more advanced technology and enlarging single chip sizes have led to the development of many large-scale monolithic accelerators with tens of billions of transistors [23], [24], [28], [55], the end of Moore's Law [35] and limited photomask size pose significant challenges to further transistor integration.

Chiplet technology, using advanced packaging to combine small functional dies, offers a promising solution to overcome these limitations and enable continuous transistor integration. Chiplet-based DNN inference accelerators, such as Simba with 36 dies [46], have emerged. However, this technology introduces new challenges for architectural design and DNN mapping for large-scale chiplet accelerators, which are outlined below:

For architecture design, the main challenge is determining the optimal chiplet granularity. While chiplet technology improves area limits and yield, it introduces higher packaging expenses and D2D interconnection costs. D2D interconnections are more energy and area-intensive, but provide lower bandwidth than on-chip lines. All the above adverse effects are collectively referred to as *Chiplet Costs*. Consequently, A trade-off arises between using more smaller chiplets for better yield and fewer larger chiplets for lower *Chiplet Costs*. Balancing this trade-off remains an unsolved challenge.

For DNN mapping, the main challenges stem from the larger scale enabled by chiplet technology, and the costly D2D links. For the first challenge, maintaining high utilization and energy efficiency becomes increasingly difficult with the growing scale of accelerators. LP mapping, in which multiple layers are spatially mapped onto the accelerator, is widely employed by large-scale accelerators in both academia [15], [45], [46], [57] and industry [21], [52], [55] to relieve the challenge. The core of LP mapping is spatial mapping (SPM), which determines which part of which layer is allocated to which core, and significantly impacts the performance and energy efficiency of large-scale accelerators. Despite the importance of LP SPM, most current strategies are still heuristic [15], [21], [46], [55], [57]. The problems and optimization space of LP SPM have not been clearly defined, exhaustively explored, or fully understood. This limitation constrains the ability to fully leverage optimization opportunities in LP mapping, which becomes increasingly important as the scale of accelerators and the structural complexity of DNNs increase. The second challenge lies in that D2D links tend to consume more energy and provide lower bandwidth than on-chip lines. Therefore, devising spatial mapping strategies that can automatically reduce D2D communications is vital for enhancing chiplet accelerator performance and efficiency, and fully harnessing the benefits provided by chiplet technology. However, there is a noticeable absence of such approaches in the existing literature.

The above analysis reveals that the employment of chiplet technology introduces intricate trade-offs and challenges.

2378-203X/24/$31.00 ©2024 IEEE
DOI 10.1109/HPCA57654.2024.00022

Thus, maximizing the benefits of chiplet technology and minimizing its disadvantages is crucial for developing DNN chiplet accelerators. This goal not only poses challenges for architectural design but also necessitates more efficient mapping schemes that effectively utilize larger accelerators while reducing the expensive communication costs between chiplets. To address these challenges, we have made the following contributions:

(1) We propose a layer-centric encoding method for representing LP SPM schemes in many-core chiplet DNN inference accelerators. Leveraging this encoding method, we delineate the optimization space for LP mapping, calculate its immense size, which significantly outstrips existing heuristic strategies, and analyze the latent optimization opportunities concealed within this space. These opportunities become particularly crucial under post-Moore chiplet times. To the best of our knowledge, this is **the first** work that clearly and systematically defines the optimization space of LP SPM for DNN inference accelerators.

(2) Based on the aforementioned encoding and a highly configurable and universal hardware template, we develop Gemini, a mapping and architecture co-exploration framework for large-scale DNN chiplet accelerators. Gemini includes two main engines: the Mapping Engine and the Monetary Cost Evaluator. In the Mapping Engine, a Simulated Annealing (SA) algorithm with five specifically-designed operators is developed to explore the extensive space defined by our encoding method and automatically minimize costly D2D communication. The Monetary Cost Evaluator assesses the MC of accelerators with varying architectural parameters. To the best of our knowledge, Gemini is **the first** framework to jointly explore the optimization space of mapping and architecture for large-scale DNN chiplet accelerators, considering not only energy consumption and performance but also MC. **The Gemini framework is open-sourced at https://github.com/SET-Scheduling-Project/GEMINI-HPCA2024.**

(3) We utilize Gemini to reveal several intriguing insights about utilizing chiplet technology in architecture design and mapping DNN workloads under chiplet scenarios as follows. (1) With the support of advanced mapping, moderately partitioning an accelerator into chiplets can reduce MC with nearly no loss of performance and energy efficiency. However, overly fine-grained chiplet partitions will simultaneously worsen MC, performance, and energy efficiency. (2) Regarding the granularity of computing cores, both energy efficiency and performance initially increase (albeit at a decelerating pace) as the granularity becomes finer (i.e., more cores). However, they experience a slight decline thereafter. Moreover, the MC tends to rise as the granularity of computing cores becomes finer, corresponding to an increase in the number of cores. (3) With the right design considerations and aided by the Gemini framework, employing a single chiplet for multiple accelerators becomes an effective approach for DNN inference accelerators. However, we also demonstrate the limitations of a "one-size-fits-all" method, particularly highlighting the impracticality of designing a small-scale chiplet intended to cater to a very wide spectrum of computational power requirements across diverse scenarios. (4) We study the property of spatial mapping, and find that rather than allocating chiplets for each layer in a gathered manner, gathering clusters with heavy data transfer together is more important and beneficial.

(4) Compared to the SOTA Simba architecture with Tangram SPM, Gemini's co-optimized architecture and mapping, on average, achieve 1.98× performance and 1.41× energy efficiency simultaneously across various DNNs and batch sizes, with only 14.3% increase in MC.

II. Background and Motivation

A. Trade-offs Introduced by Chiplet

Chiplet technology is fundamentally an advanced packaging solution. It integrates multiple dies (chiplets) onto a substrate, which could be an organic substrate [4], [36] or a silicon interposer [19], [59]. This integration is achieved through high-density interconnections, allowing the dies to function collectively as a single chip. This technology offers numerous benefits, but it also incurs additional costs. In this section, we discuss the trade-offs of employing chiplet technology.

As shown in Fig. 1, the advantages of introducing chiplets are fourfold. Firstly, it increases the overall yield of a chip. By partitioning a large-scale monolithic chip into smaller chiplets, the overall yield of the chip can be significantly improved. For example, at the 7nm technology node, the yield of an 800 mm^2 chip and a 200 mm^2 chip are approximately 18% and 75%, respectively [13]. Second, it extends the area limit of a chip, as advanced substrates (e.g., organic substrates [68] or interposers [19]) have a larger area limit than monolithic chips (858mm^2 [19] of a reticle). Third, it enables heterogeneous integration. Unlike logic circuits, analog circuit IPs do not significantly benefit from the performance and density improvements brought about by technological advancements [4], [16], [37], [53]. Therefore, manufacturing logic circuits with advanced technologies while producing various IO-functional analog IPs with more outdated processes can save on the expensive manufacturing and design and IP-related costs associated with advanced technologies [4], [16], [36], [37]. Fourth, the ability to repurpose a single chiplet

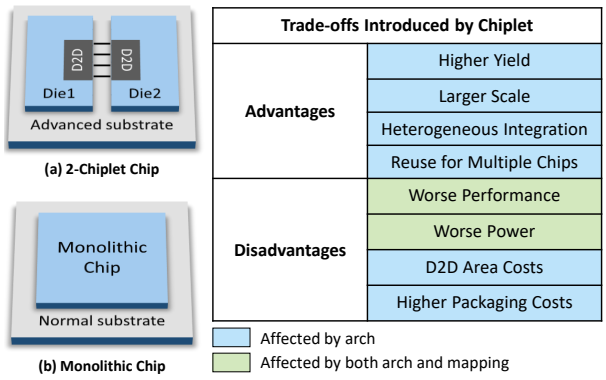

Fig. 1. Trade-offs Introduced by Chiplet

for developing multiple computational chips, each differing in scale or targeted application, is a significant advantage of chiplet technology. This approach can substantially reduce the enormous Non-Recurring Engineering (NRE) costs and the time traditionally associated with developing different chips for each scale and scenario. A notable success story in this regard is AMD's Zen series CPUs [4], [36]. While the advantages of chiplet technology are not the main focus of our research, given our primary emphasis on the architecture of an accelerator designed for a specific scenario, we do include a brief case study on this topic in Sec. VII-B for comprehensive understanding.

Fig.1 also depicts the fourfold disadvantages of chiplets, all resulting from D2D interfaces. First, they increase energy consumption, as D2D links consume several to dozens of times more energy than the less than 0.1 pJ/bit data transfer cost with on-chip lines [5], [43], [47], [48]. Second, limited inter-chiplet communication bandwidth may decrease performance. Compared to sufficient on-chip interconnect resources, interconnects between chiplets have limited bandwidth due to the limited number of IO pins available around each chiplet. Third, D2D interfaces require more area, as they need a specific analog PHY and controller, unlike on-chip lines, which occupy almost no silicon area. Fourth, the need for massive interconnections between chiplets increases packaging substrate costs, as advanced packaging scenarios require organic substrates with dozens of layers or silicon interposers, compared to the basic fan-out substrates sufficient for monolithic chips.

All the trade-offs above influence three facets of a chip: energy consumption, performance, and **MC**. It's particularly important to emphasize that in the chiplet era, solely considering the silicon chip area is inadequate for evaluating a chip's MC. Factors such as yield and packaging costs also need to be taken into account. Consequently, designing an efficient DNN chiplet accelerator requires carefully balancing among these trade-offs. However, existing works fall short in considering all these trade-offs simultaneously. Some works [13], [50] focus on studying the MC of chiplet-based chips, but do not consider their architectural details and the corresponding influence on performance and energy consumption. NN-baton [53] focuses on optimizing energy consumption and performance for small-scale DNN chiplet accelerators but does not consider the trade-offs on MC. Furthermore, NN-baton's ring-based template and layer-sequential mapping strategies limit its applicability to small-scale accelerators and restrict its scalability.

The significant potential and intricate trade-offs of applying chiplet technology to DNN inference accelerators, alongside the current research void, inspire us to create a framework that co-explores architecture and mapping for these accelerators.

B. Mapping Challenges

While chiplet technology facilitates the construction of larger-scale accelerators, translating theoretical computing power into actual computing performance presents an escalating challenge. A significant portion of existing research is focused on optimizing layer-sequential (LS) mapping for small-scale accelerators [14], [18], [20], [26], [29], [30], [33], [41], [58], [62].However, with the increase in accelerator scale, LS mapping demonstrates limited scalability [3], [15], [46], [66], whereas LP mapping exhibits a higher potential. For instance, Simba employs a naive LP mapping for some blocks of layers in ResNet-50, achieving approximately a $1.7\times$ throughput improvement over its original LS approach.

DNNs can be viewed as a Directed Acyclic Graph (DAG), where each layer is a node. In LP mapping, a graph (or subgraph) can be mapped onto a core array simultaneously, where different core groups compute different layers. The on-chip interconnect is responsible for transmitting the feature maps of layers that share dependencies. SPM determines which core specifically computes which part of which layers. As shown in Fig. 3, the top-left corner shows a DNN DAG containing two layers. In LS mapping, all six cores are used to compute each layer one by one, whereas in LP mapping, some of the six cores are used to compute the first layer, and the remaining cores are used for the second layer. The feature maps between the two layers can be transferred via the on-chip network without the need to access DRAM.

While LP mapping presents significant potential, it is less extensively researched than LS mapping. Most existing studies employ techniques such as fine-grained pipelining [3], [15] and temporal merging of some layers or workloads [7], [66] to reduce the imbalance across different pipeline stages and mitigate filling and draining overheads in LP. However, with regard to the SPM part, these studies do not offer specific optimizations and directly adopt heuristic stripe-based SPM strategies, which assign each layer to a consecutive and rectangle-shaped group of cores. The problem and its corresponding optimization space for LP SPM have not been clearly defined, let alone thoroughly explored or understood. It is worth noting that as most existing works also have an SPM optimization stage, the optimization method proposed in this work can easily integrate with existing methodologies.

Another significant challenge posed to SPM is mitigating the adverse effects on energy consumption and performance introduced by chiplet technology (Fig. 1). Fundamentally, these negative influences stem from the integration of high-energy-cost and lower-bandwidth D2D links into the chip. Consequently, this challenge primarily revolves around reducing D2D communication automatically for various workloads, an aspect that is not considered and optimized by existing heuristic LP SPM strategies [7], [15], [66].

III. Scalable Hardware Template

In this section, we introduce the architecture of our universal and configurable hardware template, which is created by extracting common features from existing chiplet accelerators [46], [52] and large-scale accelerators [14], [15], [21], [55], [61].

Overall Architecture: As shown in Fig. 2(a), the proposed template comprises two distinct types of chiplets: IO chiplets and Computing chiplets. A mesh NoC interconnects all computing cores in all Computing chiplets and the controllers

within IO chiplets, allowing for arbitrary core-to-core, core-to-DRAM, and DRAM-to-core communication. For inter-chiplet communication, the D2D transmission (TX) within a chiplet independently encodes the data and forwards it to the corresponding D2D reception (RX) in another chiplet. The D2D RX decodes the data and proceeds with NoC transmission. This inter-chiplet communication is fully automatic and transparent to both the source and the destination.

This heterogeneous architecture allows for arbitrary numbers of IO and computing chiplets, ensuring the scalability of the entire template. Otherwise, if we equip each computing chiplet with IO-related PHY and controller would occupy the chip's edge area and IO pins, which could affect the routing of D2D links and, consequently, the system's scalability.

Our hardware template and the corresponding hardware model in Gemini Framework (see Sec. V) are not limited to supporting only mesh topology. They can be easily modified to support various topologies (demonstrated in Sec. VI-B). However, in order to ensure the signal integrity of the D2D links, there are limitations on the interconnect distance and cross-linking, which restricts certain topologies like torus and butterfly. Therefore, it is preferable to maintain a point-to-point parallel interconnect, as shown in Fig. 2. Considering these factors and the fact that most existing tiled accelerators [21], [46], [52], [64] currently adopt mesh interconnect networks, we default to using the mesh topology in this paper.

Computing Chiplet Architecture: Each Computing Chiplet contains an arbitrary number of computing cores interconnected by a mesh NoC. To enhance the scalability of this template, D2D interfaces are placed around the Chiplet, whose number is equal to the number of computing cores on each side. This arrangement enables the Computing Chiplet to form a larger-scale mesh with other Chiplets. In the example illustrated in Fig. 2(a), there are four cores on each side, so we place four D2Ds on each side of the Computing Chiplet.

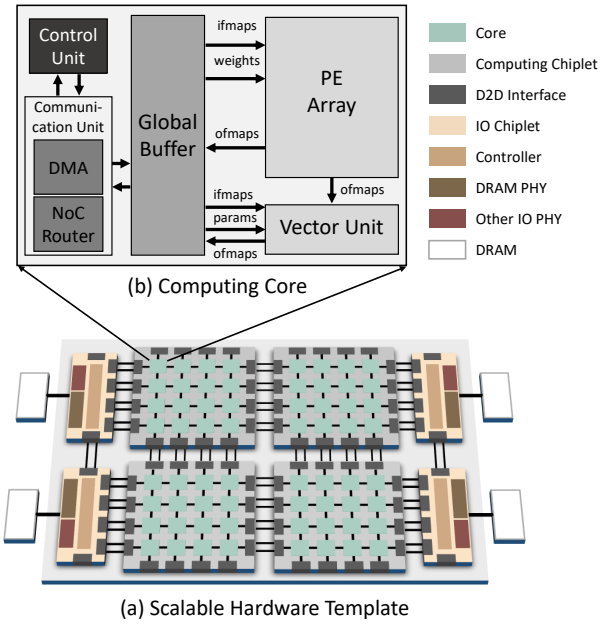

Fig. 2. Architecture of Scalable Hardware Template

The computing cores are the key components responsible for performing calculations in the entire accelerator, and their architecture is shown in Fig. 2(b). The communication unit comprises DMA and router, facilitating communication with other cores and DRAM. The control unit is primarily responsible for managing computation tasks based on statically-compiled instructions and task progress information, as well as managing the reception and transmission of data or messages to and from external cores or DRAMs. The global buffer (GLB) of each core is globally visible in the entire accelerator. Every core can read data from or write data to the GLBs of other cores, provided the data is valid, or the address can be written to. The PE array and the vector unit are responsible for computing the General Matrix Multiply (GEMM)/Convolution (Conv) and vector/scalar operators, respectively. Each time, the PE array reads a workload tile's weight and input feature maps (ifmaps) from the GLB. The resulting ofmaps/partial sum (psum) can be directly written back to the GLB or post-processed within the vector unit (such as Batch Normalization and ReLU operations). Simultaneously, the vector unit can be independently invoked to compute vector/scalar operators.

IO Chiplet Architecture: The IO Chiplet is equipped with an array of IO functionalities, enabling interactions with DRAM, host systems, or other input sources (e.g., cameras). All input data from host systems or alternative input sources are first loaded into DRAM, and then can be loaded and processed by computing cores. The DRAM controller is also connected to multiple routers within the entire mesh NoC to match the bandwidth of the DRAM and the network, ensuring the full utilization of DRAM bandwidth.

Configurable Parameters: Our template features excellent configurability, offering a range of adjustable architectural parameters. These parameters include NoC bandwidth, D2D bandwidth, total DRAM bandwidth, the total number of cores in the X-direction (e.g., 8 in Fig. 2), the total number of cores in the Y-direction (e.g., 8 in Fig. 2), the number of chiplet divisions in the X-direction (X_{Cut})(e.g., 2 in Fig. 2), the number of chiplet divisions in the Y-direction (Y_{Cut})(e.g., 2 in Fig. 2), the number of MACs in the PE array within a single core, and the size of the GLB per core. It is worth mentioning that the microarchitecture of the PE array and its corresponding dataflow have been extensively studied in existing works [8], [18], [25], [29], [53], [58], [65]. Therefore, in this work, the PE array adopts the classic NVDLA architecture [39], [58] and corresponding dataflow to maintain a fair comparison with the baseline, Simba. ***Of course, NVDLA can also be replaced by other microarchitectures with different dataflows, which are supported by our template.***

Based on this highly configurable template, we have developed matching delay, energy consumption, and MC evaluators as introduced in Sections V-B2 and V-C, which enable our comprehensive design space exploration.

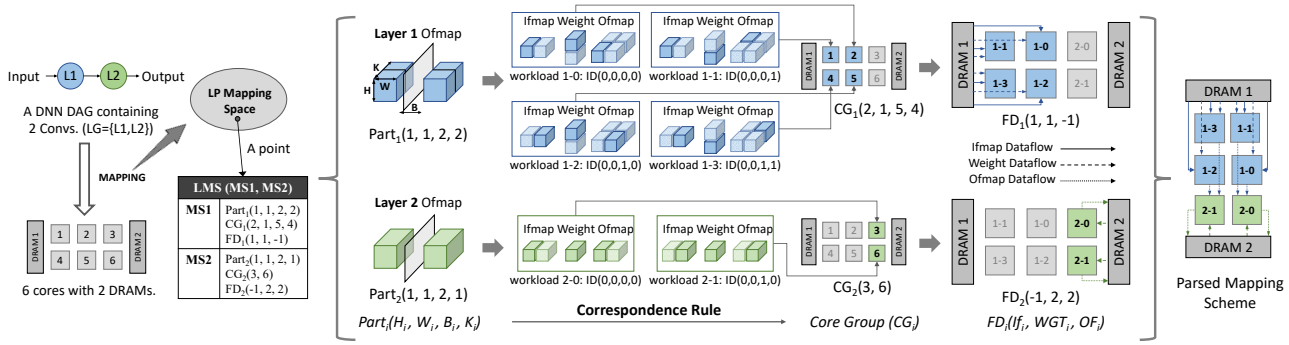

Fig. 3. Parsing an Encoded LMS in the LP SPM Optimization Space into an Actual SPM Scheme

IV. LP Spatial Mapping Encoding

A. Encoding Format and Parsing Methods

In this section, we introduce a layer-centric encoding method for describing LP SPM schemes. At a high-level, the encoded LP SPM scheme encapsulates two key aspects of information: (1) the allocation and partition of each layer to specific cores for computation, and (2) the data sources and destinations for the workload on each core. This encoding method exhibits great generality and can adapt seamlessly to diverse NoC topologies and core microarchitectures (demonstrated in Sec. VI-B2). Subsequently, we provide a comprehensive description of our encoding format and the corresponding parsing method.

Consider an N-layer DNN DAG that needs to be spatially mapped onto an accelerator in an LP manner. This accelerator features a core group CG comprising M cores and D DRAMs. The layers in the DAG form a Layer Group (LG). For the example in Fig. 3, the DAG consists of two Convs ($Layer_1$ and $Layer_2$), forming a layer group with these two layers. The accelerator in this example has a core group containing six cores and two DRAMs.

In our encoding format, an **LP Spatial Mapping Scheme** (LMS) of a layer group consists of the **Mapping Scheme** (MS) for each layer within it. The MS of layer i has three attributes: Partition ($Part_i = (H_i, W_i, B_i, K_i)$), Core Group ($CG_i = (C_{id_{i,1}}, C_{id_{i,2}}, ..., C_{id_{i,nc_i-1}})$, where nc_i is the number of cores in CG_i), and Flow of Data ($FD_i = (IF_i, WGT_i, OF_i)$, $-1 \leq IF_i, WGT_i, OF_i \leq D$). The left side of Fig. 3 shows the LMS of a layer group containing two layers and the MS of each layer.

$Part_i$ partitions $layer_i$ along each dimension of the four-dimensional output cube into approximately equal nc_i parts for each core in CG_i. As shown in Fig.3, the four dimensions are Batch (B), representing the number of samples processed at a pipeline stage; ofmaps channel, which is also the weight kernel (K); ofmaps Height (H); and ofmaps Width (W). B_i, K_i, H_i, and W_i represent the partition of the corresponding dimension. Based on this ofmaps partition scheme, the partition schemes for ifmaps and weights can be uniquely determined based on the features of different types of layers. For the example in Fig.3, $layer_1$ is a Conv, with B and K equal to 2 and 2, respectively (other dimensions are not partitioned in this example). The example shows how $Part_1$ partitions $layer'_1s$ ofmaps (on the left of the arrow) and how the partition scheme of the ofmaps deduces the corresponding ifmaps and weight partition schemes (on the right of the arrow).

CG_i contains the cores dedicated to computing $layer_i$. CG_i is ordered ($(C_1, C_2) \neq (C_2, C_1)$). Each core of it can be an arbitrary core in CG. As shown in Fig. 3, CG_1 is $(2, 1, 5, 4)$.

We then establish a Correspondence Rule to map each partitioned workload (PW) of $layer_i$ to a corresponding core in CG_i. First, we assign each PW a unique 4-dimension ID based on its location in the partitioned ofmaps cube ($PW = (h, w, b, k), h \in [0, H_i), w \in [0, W_i), b \in [0, B_i), k \in [0, K_i)$. Then based on the 4-dimension ID, we transform it into a numerical ID, which equals to $h \times W_i \times B_i \times K_i + w \times B_i \times K_i + b \times K_i + k$. The numerical ID ($NID$) of each partitioned workload corresponds to the $(NID + 1)$th core in CG_i it is assigned to. For example, in Fig. 3, the top left subfigure of the four is $layer_1$'s first part of both batch and ofmaps channel, and H and W of $layer_1$ are not partitioned; thus, its four-dimension and numerical ID are $(0, 0, 0, 0)$ and 0, respectively. It is mapped to the first core (C_2) in CG_1.

FD_i represents the data sources of $layer_i$'s ifmaps (IF_i) and weights (WGT_i), and the destination of $layer_i$'s ofmaps (OF_i). The flow of data can be categorized into two types: those that require explicit management (non-negative values) and those that do not require explicit management or are directly absent (-1).

The scenarios that require explicit management are as follows: (1) For ofmaps, when the subsequent layer is not in the same layer group as the current layer or when the output of the current layer is the output of the entire DNN, it is necessary to explicitly manage the temporary storage of this layer's output to a specific DRAM. (2) For ifmaps, explicit management is only required when the input of the current layer is the input of the entire DNN; otherwise, the data can be fetched from the DRAM where the previous layer's ofmaps were stored. (3) For weights, explicit management is required whenever a layer has weights. Hence, the primary concern for explicitly managing data flow is determining from which

DRAM to store or fetch data. In cases where the value is greater than 0, it represents the ID of the DRAM. Meanwhile, 0 represents a special case - interleaving, where we evenly distribute data across all DRAMs to fully and evenly utilize the available bandwidth. For example, as depicted in Fig. 3, due to $layer_1$ having the input of the whole DNN and weights, IF_1 and WGT_1 should be non-negative. FD_1 of $layer_1$ in the example is (1,1,-1), indicating that the ifmaps and weights of $layer_1$ originate from the DRAM 1.

If both layers sharing dependency are in the same layer group, the ofmaps of the previous layer and the ifmaps of the subsequent layer need not be explicitly manipulated because the destination of each part of the previous layer and the source of each part of the subsequent layer can be directly inferred based on the partition schemes and CGs of these layers. Moreover, for the layers without weights, their WGT_i value would be -1. For example, in Fig. 3, based on $Part_1$, CG_1, $Part_2$, and CG_2, the data communication dependency between cores of $layer_1$ and $layer_2$ can be deduced directly. Thus, OF_1 and IF_2 are -1.

B. Space Calculation

As shown in Fig. 3, each LMS is a point in the optimization space defined by our encoding method. The optimization space of mapping N layers onto an accelerator with a core group containing M cores and D DRAMs is considerably large and extremely complex to calculate. Therefore, we conservatively approximate its lower bound size at $m! \sum_{i=0}^{N-1} \binom{N}{i}\binom{M-N-1}{N-i-1}4^{N-i}$ schemes, where $\binom{x}{y}$ is the binomial coefficient, which equals $\frac{x!}{y!(x-y)!}$. As a comparison, the upper bound of optimization space of the SOTA heuristic Tangram is $N \cdot part(M)$, where $Part_M$ represents the total number of possible factorizations for a given integer M. We can observe that the optimization space defined by our encoding method is significantly larger than the optimization space of the Tangram heuristic. The detailed calculation procedure and tables of the optimization space under different M and N for our method and Tangram can be found at this link [2].

C. Unveil Hidden Optimization Opportunities

In this section, we will introduce the general optimization opportunities for multi-core DNN accelerators hidden within this space, as well as how these opportunities become even more significant in chiplet scenarios.

First, different $Part$ attributes of each layer impact two aspects: (1) NoC communication volume: Various partition schemes lead to different data requirements for each core, causing disparities in NoC communication volumes, even with multicast capabilities. For instance, in Fig. 3, under $Part_1$, each core requires half of the ifmaps and weight. However, if the $Part$ is changed to $(1, 1, 1, 4)$, each core needs the entire ifmaps and only 1/4 weights. (2) Intra-core optimization space: The dimension of the partitioned workload generated by distinct partition schemes differ, subsequently affecting the optimal intra-core dataflow scheme.

Secondly, the number and position of cores can vary in each CG attribute. The number of cores affects the computation time of each layer, which in turn influences the overall pipeline computation time, as the slowest stage can stall the pipeline. Core positions can significantly impact the data transmission volume and congestion levels of the NoC.

Third, different FD attributes affect the bandwidth utilization and access patterns of different DRAMs, and NoC communication. As shown in Fig. 3, the ifmaps and weights of $layer_1$ are read from $DRAM1$, while the weight and ofmaps of $layer_2$ are read from and written into $DRAM2$, respectively. In this case, the bandwidth demand and access patterns for each DRAM are not balanced, but the total NoC hops are relatively few. If all positive values in FD_1 and FD_2 change to 0 (interleaved), DRAM bandwidth usage will become more balanced. However, the total NoC hops increase since some data will interact with more remote DRAM.

V. Gemini Framework

A. Gemini Overview

Gemini is a mapping and architecture co-exploration framework for DNN inference chiplet accelerators. As depicted in the left of Fig.4, the inputs for Gemini consist of (1) Architecture parameter candidates: Each configurable architecture parameter (introduced in Sec.III) is assigned a list of candidate values; (2) Framework settings: These include the optimization goals and constraints, hyperparameters, and other relevant settings; (3) DNN models: Considering that an accelerator is often used to accelerate different DNNs under different scenarios, Gemini supports conducting DSE for n DNNs.

All architectural candidates are exhaustively explored with the optimization objective of $MC^{\alpha} \times E^{\beta} \times D^{\gamma}$, where MC, E, and D denote Monetary Cost, Energy Consumption, and Delay, and α, β, and γ denote the respective importance of MC, E, and D (performance). Note that the delay associated with small and large batch sizes can be utilized to illustrate performance in latency-sensitive and throughput-sensitive scenarios, respectively. The E and D are not only influenced by the architecture but also by the specific DNN workloads and their corresponding mapping strategies. Thus, the Mapping Engine employs a dynamic-programming-based graph partition algorithm and an Simulated-Annealing-based (SA-based) LP SPM exploration algorithm to optimize the mapping of the ith DNN onto the architectural candidate. SA algorithm is a widely-used optimization algorithm [27]. This optimization process utilizes $E_i^{\beta} \times D_i^{\gamma}$ as the optimization goal and yields the evaluation of E_i and D_i for processing the ith DNN with the given architecture parameters. Consequently, the overall Energy Consumption and Delay of the architectural candidate are determined by $(\prod_{i=1}^{n} E_i)^{\frac{1}{n}}$ and $(\prod_{i=1}^{n} D_i)^{\frac{1}{n}}$, respectively. In contrast, the Monetary Cost remains unaffected by workloads and mapping strategy, enabling the MC Evaluator to directly assess it based on the architecture parameters.

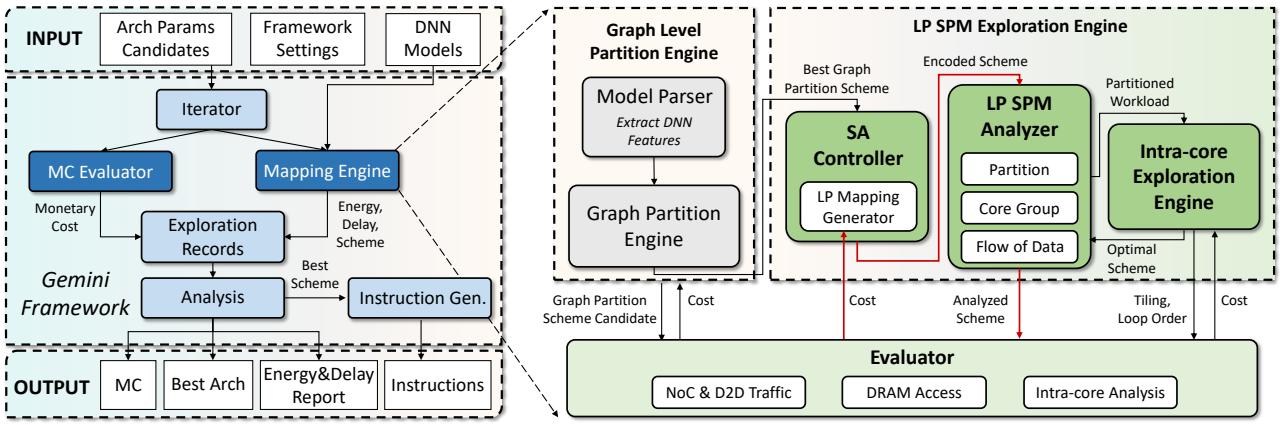

Fig. 4. Gemini Framework. The red arrow illustrates the Simulated Annealing (SA) iteration loop.

B. Mapping Engine

As shown in the right part of Fig. 4, each DNN description file is first processed by the model parser, which generates a DNN topology graph and extracts the features of each layer. Then, this information is sent to the graph partition engine, which partitions the DNN graph into layer groups. Based on the explored graph partition scheme, the LP SPM exploration engine employs an SA-based algorithm with five specifically-designed operators to explore the LP SPM optimization space defined in Sec. IV-B for each layer group.

Since the graph partitioning problem has been extensively studied [15], [22], [34], [66], [67], we employ the same DP-based graph partition algorithm as Tangram [15], which is also our baseline, to ensure a fair comparison in our experiments. This algorithm not only generates layer groups efficiently but also determines the number of samples (batch unit) processed in each pipeline stage [15].

1) LP SPM Exploration Engine: In this engine, Gemini employs an SA-based algorithm with five specifically-designed operators to explore the broad space defined in Sec. IV based on the graph partition scheme explored before.

For each layer group, the initial LP SPM scheme is obtained using a widely adopted heuristic stripe-based strategy [15], [57], [66]. Then, the SA iteration starts, and the iteration loop is represented by the red arrow in Fig.4. In each iteration, the SA controller randomly selects a layer group with a probability distribution proportional to their optimization size as calculated in Sec. IV-B. It then randomly selects one of the five operators to apply a transformation to the chosen layer group. Following this, the LP SPM Analyzer analyzes the modified scheme as introduced in Sec. IV-A. After analyzing the partition attribute of each layer, the partitioned workload will be scheduled in intra-core exploration engine, which performs exhaustive search optimization for tiling and loop reorder like many existing works [29], [41], [53], [58], [65]. Once the optimal solutions of intra-core dataflow for all partitioned layers are found, they will be sent along with the other analyzed information to the Evaluator for overall assessment. If the overall cost is lower, the change is accepted; otherwise, it is accepted with a probability which decreases as the number of iterations increases.

SA Operators: We develop five SA operators to facilitate the exploration process in SA. The operators are as follows:

OP1: Randomly select a layer and change the values in its $Part$, while still satisfying the constraints of $Part_i$.

OP2: Randomly select a layer and randomly swap two cores within its CG, which is equivalent to exchanging the workload between these two cores for a single layer randomly.

OP3: Randomly select two layers and swap two cores within their CGs, which is equivalent to exchanging the workload between these two cores for two layers.

OP4: Randomly select two layers, remove a core randomly from the CG of one layer, and add it to the CG of the other layer. After the operation, update the $Parts$ of both layers randomly to match their new CG sizes.

OP5: Randomly select a layer, then choose a non-negative item in its FD randomly, and update its value within the range of 0 to the number of DRAMs randomly.

Utilizing these operators allows each attribute to transition into any other state that fulfills the corresponding constraints through a sequence of transformations. For instance, the size of CG_1 in Fig. 3 can be modified to any number from 1 to 5 through a series of $OP4$ operations. Crucially, this ensures that any point within the LP SPM optimization space can be reached from any other (demonstration link [1]), thereby guaranteeing comprehensive exploration by the SA algorithm, and yielding near-optimal solutions.

Through our SA-based algorithm combined with our specially designed operators, Gemini not only can explore the optimization space to balance the trade-offs introduced in Sec IV-C, but also automatically optimize D2D link communication. Since D2D links tend to have smaller bandwidth and higher energy consumption, during the iterative process, if an SA operation increases the use of more D2D links, it is more likely to significantly reduce performance and energy efficiency, making it less likely to be accepted. On

the other hand, SA operations that reduce D2D link usage are more likely to be accepted. Therefore, the entire search process inherently optimizes D2D communication, which will be demonstrated in Sec. VII-C. Furthermore, this mapping technique can aid in architecture design by enabling accelerators to be equipped with lower D2D bandwidth, reducing their area overhead and reaping the benefits of chiplet-induced yield improvement while incurring only minimal performance and energy efficiency losses.

2) Evaluator: The Evaluator in the Gemini framework, which is modified based on SET [7], has two interfaces, one for intra-core evaluation and the other for global evaluation. When called by the intra-core exploration engine, the Evaluator can calculate the number of operations for each part, such as the memory access times for different-level buffers and the number of different-precision multiply-accumulate (MAC) operations. Based on this information, the total energy consumption can be calculated by summing up the number of operations for each component multiplied by the corresponding unit energy consumption. The overall computation time for the workload on the core can be determined by taking the maximum value among the MAC computation time and the data access amounts for each memory level divided by their respective access bandwidths.

Based on the LP SPM scheme analyzed by the Analyzer and the explored intra-core scheduling scheme for each partitioned layer, the Evaluator conduct the global evaluation by analyzing the data communication volume on each on-chip network link and D2D link, access patterns of DRAM, memory access times for different-level buffers within each core, and operation counts for various-precision computation units. Subsequently, a simulator assesses the delay of the DNN with specific batch size on this accelerator. The energy consumption can be calculated by summing up the number of operations for each component in the accelerator, multiplied by their respective unit energy consumption. It is worth mentioning that the energy consumption of NoC routers is predominantly attributed to the input buffer and crossbar components. Therefore, the per-flit energy consumption of NoC routers does not vary significantly across different traffic patterns and can be considered a constant value [60]. The energy consumption calculation for D2D links can be categorized into two types. The first type is for the clock-embedded D2D, where the clock signal does not have a dedicated channel and needs to be recovered from the data. A typical example is SerDes [47]–[49]. For this type of D2D, almost the same amount of power is consumed regardless of whether data is being transmitted. Therefore, its energy consumption is calculated as *Number of D2Ds* $\times$ *Power per D2D* $\times$ *Latency*. The second type is the clock-forwarding D2D, which has a separate clock channel, such as GRS [42], [43], and UCIe [54]. This type of D2D can enter a low-power state when not transmitting data. Hence, its energy consumption is calculated as *D2D Communication Volume* $\times$ *Unit D2D Communication Energy Consumption*, similar to on-chip networks. Since the baseline of this work is Simba with GRS links, the second model is also the default model in the experimental to guarantee a fair comparison.

C. Monetary Cost Evaluator

MC Evaluator can assess the production cost of different architecture candidates, which mainly includes chiplet silicon cost, DRAM cost, and packaging cost.

The evaluation of the silicon area for all chiplets ($Area_tot = \sum_{i=1}^{n} Area_{Die_i}$) serves as a fundamental cornerstone within the MC Evaluator. The chip silicon cost and packaging cost are directly influenced by this area evaluation, as shown in the following paragraphs. Each chiplet's area is determined by the summation of the areas of its constituent modules. In this work, the area of analog IPs, such as PCIe PHY, DDR PHY, and D2D PHY, is obtained directly from the corresponding datasheets. For the logic modules, their area estimation is based on the Verilog code and evaluation process employed during the development of our chip.

The silicon cost of a die (chiplet) is $Area_{Die_i}/Yield_{die_i} \cdot C_{silicon}$, where $Yield_{Die_i} = Yield_{unit}^{Area_{Die_i}/Area_{Die_{unit}}}$ [13]. $Yield_{unit}$ represents the yield per unit area ($Area_{Die_{unit}}$), where $Yield_{unit}$ under 12nm in this paper is assumed to be 0.9, and $Area_{Die_{unit}}$ is set to 40 mm^2.

The DRAM cost is $\lceil DRAM_{bw}/Unit_{bw} \rceil \cdot C_{DRAM_{Die}}$, where $Unit_{bw}$ and $C_{DRAM_{Die}}$ represent the bandwidth provided by each DRAM die and the MC of each DRAM die, respectively. The $Unit_{bw}$ and $C_{DRAM_{Die}}$ used in this work (GDDR6) are $32GB/s$ and \$3.5 [12].

The packaging cost equals $(Area_{tot} \cdot f_{scale})/Yield_{package} \cdot C_{package}$. Because the substrate requires a larger area than the total area of all chiplets to accommodate IO fanout and interconnect wiring functions, there is an empirical scaling factor f_{scale} to calculate the substrate area based on total silicon area [13]. $C_{package}$ represents the monetary cost per unit area of the substrate. The $C_{package}$ for different substrates varies. For instance, in the case of an organic substrate, without adopting chiplet technology, a standard fan-out substrate is relatively inexpensive ($0.005\$/mm^2$). However, when chiplet technology is employed, the price increases due to the need to support high-density interconnects. $C_{package}$ varies across different area ranges. Larger substrate areas require more intricate manufacturing processes, resulting in higher C_{package}.

D. Future Work Grounded on Gemini

The field of DNN chiplet accelerators is still emerging, with limited research available. Gemini offers a fertile ground for further exploration. In this section, we outline examples of promising future work in the realms of mapping and architectural design that can be pursued using Gemini. Our intention is to underscore the value of Gemini while also fostering community development.

At the mapping level, a promising direction is to co-explore the SPM optimization dimension with other optimization dimensions, such as the graph-level composite spatial-temporal dimension defined by SET. Jointly exploring these dimensions could potentially yield better solutions, and it would also be valuable to investigate the interplay between

these different optimization dimensions On the architectural front, the heterogeneity of chiplet presents another compelling area for research. Questions around scheduling LP mapping on heterogeneous chiplets and, reciprocally, exploring architectural designs for heterogeneous accelerators in the context of LP mapping are of particular interest.

VI. Evaluation

A. Experiment Setup

1) DSE Configurations: In this work, we conduct DSE on DNN accelerators with 72 TOPs, 128 TOPs, and 512 TOPs to show Gemini advantages and gain a deeper understanding of the design space for DNN chiplet accelerators and the use of chiplet technology. We choose 72 TOPs instead of 64 TOPs to enable comparison with the SOTA Simba architecture with 72 TOPs [46]. Table I presents the DSE parameters. By keeping the total computing power constant, we determine the number of cores based on the MAC/Core configurations. To maintain the core array's length and width as close as possible, we arrange the cores accordingly. For instance, with 36 cores, we configure them in a 6×6 arrangement, while for 18 cores, we adopt a 6×3 configuration. The values of X_{Cut} and Y_{Cut} must be a factor of the corresponding number of cores on edge; otherwise, the candidate is deemed invalid.

We choose TSMC 12nm as the default process, which is also widely used by many commercial accelerators [23], [31], [38]. We choose organic substrate as the default packaging substrate, as it is used in many successful commercial [4], [36], [37] and academic designs [46]. The default operating frequency is set to 1GHz. For D2D, we use GRS [43], [46] as default in experiments to guarantee a fair comparison with Simba.

For the sake of brevity, the explored architectural parameters are presented as (*Chiplet Number, Core Number*, $DRAM_{BW}$, NoC_{BW}, $D2D_{BW}$, $GBUF/Core$, $MAC/Core$).

The DSE adopts a batch size of 64, targeting a throughput-driven scenario, as defined by the MLPerf benchmark [44]. Although the exploration process solely utilizes these two DNN models with a batch size of 64, we also evaluate additional DNNs with batch size 1 (latency-centric scenarios) on the explored architecture, demonstrating its broad applicability to other scenarios. We set the default optimization objective of DSE as $MC \cdot E \cdot D$ (as introduced in Sec. V-A). The default workload is set as Transformer [56] since it is prevalent and widely used in various scenarios such as image [11] and language processing [6], [10], [40].

TABLE I
DSE Parameters. The parameters marked in *italics* and bold are exclusively used by DSE for 72TOPs and 128/512 TOPs accelerators, respectively.

Accelerator Config.	X_{Cut}	*1, 2, 3, 6* (**1, 2, 4, 8**)
	Y_{Cut}	*1, 2, 3, 6* (**1, 2, 4, 8**)
	$DRAM_{BW}$	0.5, 1, 2 GB/s per TOPs
Network Config.	NoC_{BW}	*8*, 16 ,32, 64, **128** GB/s
	$D2D_{BW}$	NoC/4, NoC/2, NoC
Core Config.	$GBUF/Core$	256, 512, 1024, 2048, 4096, **8192 KB**
	$MAC/Core$	*512*, 1024, 2048, 4096, **8192**

2) DSE Time: The DSE time increases with the increase in target computing power. For example, the DSE for 72 TOPs and 512 TOPs use 80 threads and 100 threads, respectively, and run for 2280s and 23907s on an Intel Xeon Platinum 8260 server.

3) Workloads: To fully demonstrate the advantages of our architecture and mapping co-exploration, we conduct the comparison with the baseline over a wide range of DNNs, including ResNet-50 (RN-50) [17], ResNeXt (RNX) [63], Inception-ResNet (IRes) [51], PNasNet (PNas) [32], and Transformer (TF) [56]. ResNet-50 and ResNeXt are selected due to their use of classic residual structures, which are prevalent in many DNNs. Inception-ResNet-v1 and PNASNet represent DNNs with more intricate dependencies. Due to the reasons mentioned above, we mainly use Transformer as a representative to showcase various observations and insights, in addition to overall comparisons.

4) Baseline: The baseline here principally comprises two components: baseline architecture and baseline mapping. For the architecture baseline, we adopt Simba [46] and optimize it accordingly. Since the Simba accelerator is a test chip lacking DRAM, the configuration of its on-chip buffer is also impacted. Consequently, we initially equip it with I/O Dies and furnish a total DRAM bandwidth of 2 GB/s per TOPs. Concurrently, the on-chip buffer configuration is determined according to the Simba-series paper [58], which explores the architecture of an NVDLA-style core. In particular, the GBUF is allocated 1024 KB per core. The remaining configurations align with the specifications in the original Simba paper [46]. In experiments, the baseline architecture is abbreviated as **S-Arch**, while the architecture explored by our Gemini framework is denoted as **G-Arch**. Additionally, to demonstrate the universality of Gemini, we conduct an exploration with a hardware template modified to adopt a folded torus NoC topology. We compare the explored architecture and mapping scheme with an accelerator that employs the Tenstorrent Grayskull architecture parameters [55] (**T-Arch**) and utilizes Tangram mapping. This comparison serves to validate the effectiveness of the Gemini (Sec. VI-B2).

Although Simba has its own layer-sequential mapping [46], this mapping performs poorly on such a high-computing-power chip [15], [66]. Therefore, we choose the SOTA Tangram LP mapping strategy as the baseline mapping strategy (abbr., **T-Map**). The mapping scheme explored by Gemini framework is abbreviated as **G-Map**.

B. Overall Comparison

1) Compared to S-Arch with T-Map: As illustrated in Fig. 5, G-Arch mapped with G-Map achieves a 46.8% reduction in delay and a 28.8% reduction in energy consumption across five DNNs and two batch sizes when compared to S-Arch mapped with T-Map. This improvement is attained with only

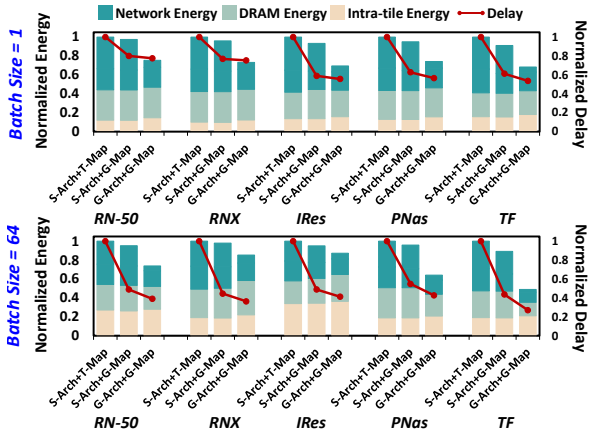

Fig. 5. Overall Comparisons among G-Arch, G-Map, S-Arch, and T-Map. energy consumption and delay in each comparison are normalized to overall baseline **S-Arch+T-Map**

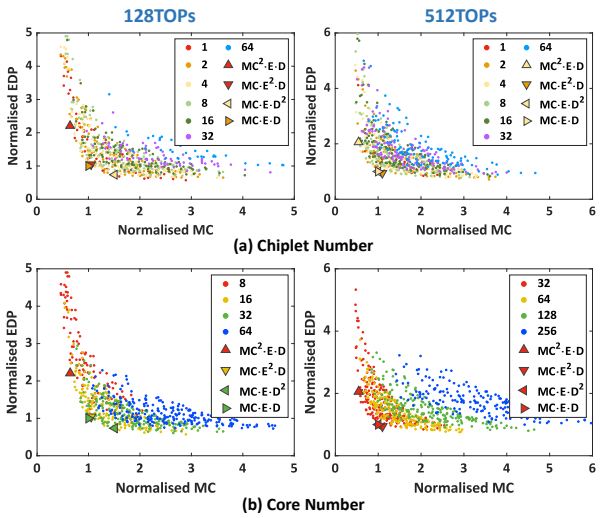

Fig. 6. EDP and MC of the Architecture Candidates in the Design Space for 128 and 512 TOPs Accelerators. Different colors represent various categories of architectural parameters. To provide a clearer depiction of trends, only the top 50% of each category is plotted. The EDP and MC of each point are normalized to the counterpart of the best arch under $MC \cdot E \cdot D$. We use triangles pointing in different directions to represent the globally optimal architectures under different optimization objectives. The workload is Transformer.

a 14.3% additional MC, demonstrating the value of Gemini co-exploration. Moreover, by merely employing G-Map on the S-Arch, a significant reduction delay and energy consumption can be achieved compared to T-Map. The optimization brought up by Gemini mapping demonstrates that exploring the vast space defined by our encoding method can indeed achieve a better balance of the trade-offs introduced in Sec. IV-C. The explored G-Arch is (2, 36, 144GB/s, 32GB/s, 16GB/s, 2MB, 1024). Compared to S-Arch, the number of Chiplets in G-Arch is significantly reduced, while the bandwidths of NoC and D2D are increased, and the GBUF capacity is doubled. The reduction in the number of Chiplets leads to a significant decrease in the proportion of D2D links across all links, resulting in a substantial reduction in network energy. With improved interconnect bandwidth and on-chip storage resources, performance and energy efficiency improvements are natural. However, the interesting point is that these resource improvements only result in a 14.3% increase in MC. This is mainly because, under S-Arch, there are too many Chiplets, and an excessive amount of chip area is used for D2D interfaces (nearly 40%). In G-Arch, this area is converted into interconnect bandwidth and on-chip storage resources. However, it is worth noting that Simba serves as an academic test chip. As a pioneering work in introducing Chiplets into DNN accelerators, Simba has already showcased many features of Chiplet-based inference accelerators. Given that Simba is an academic demo with limited chiplet area and scale (6 mm^2 [68]), its suboptimal chiplet granularity is understandable. Further insights regarding the optimal choice of chiplet granularity will be discussed in Section VII-A1.

2) Compared to T-Arch with T-Map: We compare the architecture and mapping scheme explored by Gemini with a 120-core monolithic accelerator that utilizes the same architectural parameters as Tenstorrent Grayskull (Core array size, MAC & GBUF per core, and DRAM bandwidth) [55] and employs Tangram mapping. Gemini's explored architecture is (6, 60, 480 GB/s, 64GB/s, 32GB/s, 2MB, 2048). Compare with T-Arch with T-Map, G-Arch with G-Map achieves 1.74× performance and 1.13× energy efficiency, and reduces 40.1% MC, which demonstrates the effects and universality of Gemini.

VII. Discussion

A. Design Space Exploration

1) Chiplet Granularity: As illustrated in Fig. 6(a), the optimal number of chiplets for 128 TOPs and 512 TOPs, under the four different objectives, ranges from 1 to 4 and 2 to 4, respectively. Moreover, it is evident from both DSEs that an excessively fine-grained chiplet granularity negatively impacts MC, energy consumption, and performance. Combining it with the analysis in comparing G-Arch and S-Arch in Sec. VI-B, we can achieve an insight: ***Partitioning DNN accelerators into chiplets with moderate granularity can effectively strike a balance among MC, energy consumption, and performance. Conversely, an excessively fine-grained approach to chiplet partitioning can have a negative impact on all three metrics—performance, energy consumption, and MC—simultaneously.***

2) Core Granularity: In addition to the granularity of chiplets, the granularity of cores is also an important topic in many-core architectures. Particularly in the context of LP mapping, the trade-offs associated with varying core granularities have yet to be explored. This section aims to address this gap in the literature.

As shown in Fig. 6(b), there is a clear trend of increasing MC as the number of cores increases. This is primarily because, to improve overall performance and energy efficiency, it

is necessary to increase network bandwidth or buffer capacity per core. The more cores there are, the more these resources multiply, leading to greater area overhead. However, from the perspective of EDP, it initially decreases with an increase in the number of cores, then rises again later. For example, in the 128 TOPs scenario, the EDP for the yellow and green points is better than that for the red and blue points. This intriguing phenomenon warrants further analysis to understand the underlying trade-offs.

A main advantage of LP Mapping is to reduce DRAM accesses, which is a key factor contributing to improvements in both energy consumption and performance. Under LP Mapping, an increased number of cores facilitates longer pipelines, allowing for the simultaneous processing of more layers. This, in turn, enables more dependency data to be transferred and processed on-chip, thereby reducing costly DRAM accesses. However, not all dependencies have the same amount of data. Layers involving links with larger data volumes will be prioritized for simultaneous processing. Therefore, the reduction in DRAM access due to the increase in the number of layers processed simultaneously diminishes at a slower rate. Moreover, as the pipeline depth increases, so does the utilization loss due to the filling and draining phases [15], [66]. In summary, a longer pipeline is not necessarily better. The diminishing benefits from extending the pipeline and the escalating costs eventually reach a point of equilibrium, which represents the optimal pipeline length. As evidence, in Fig. 7, the number of cores for the optimal solutions under the four different objectives, from left to right, are 16, 8, 32, and 32. Correspondingly, the average number of layers processed simultaneously for each are 5.4, 4.1, 10.2, and 8.1. As shown in Fig. 7 Left, though DRAM access continuously decreases as the number of cores increases from 8 to 32, the rate of reduction is much faster when transitioning from 8 cores to 16 cores (48.0%) compared to the transition from 16 to 32 cores (average 19.4%). Moreover, the average number of layers processed simultaneously by the optimal 64-core architecture (the best among all 64-core configurations under the target of $MC \cdot E \cdot D$) is 9 layers, which does not continue to increase with the number of cores, indicating the striking of the balance.

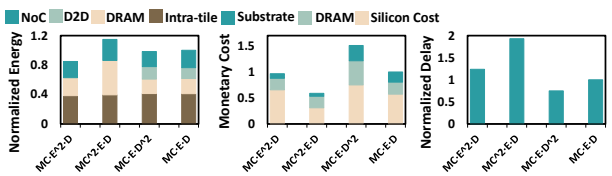

Fig. 7. Comparative Analysis of Energy Consumption, MC, and Delay for Optimal Architectures Explored under Four Different Optimization Objectives. In the left-to-right order, they are: (1, 16, 128GB/s, 32GB/s, None, 4MB, 4096), (1, 8, 128GB/s, 32GB/s, None, 4MB, 8192), (4, 32, 256GB/s, 64GB/s, 32GB/s, 2MB, 2048), and (2, 32, 128GB/s, 32GB/s, 16GB/s, 2MB, 2048). All data are normalized to $MC \cdot E \cdot D$.

B. Reuse A Single Chiplet for Multiple Accelerators

One significant advantage of utilizing chiplets in the industry lies in their reusability across accelerators of varying scales. This approach has already been successfully employed in CPUs [4], [36], [37]. Despite its promise, this potential benefit has yet to be explored within the realm of DNN inference accelerators. This section aims to bridge this gap using Gemini Framework.

Figure 8 reveals that the performance of accelerators constructed with Simba chiplets is poor for both 128 TOPs and 512 TOPs configurations, with the latter faring even worse. ***This example illustrates the failure of a "one-size-fits-all" approach, highlighting the impracticality of designing a small-scale chiplet to cover an extensively wide range of scenarios***. Additionally, although using chiplets from the best 128 TOPs accelerator in the 512 TOPs accelerator yields better results than Simba, the overall performance is still unsatisfactory, and vice versa. This example, in conjunction with the Simba case, collectively illustrates that ***directly repurposing chiplets from one computing platform to build another is ill-advised and frequently results in less-than-ideal outcomes***.

To fully harness the potential of "Chiplet Reuse for Multiple Accelerators", we have enhanced Gemini to enable DSE for multiple computational power levels concurrently. In this approach, Gemini strategically organizes the chiplets of each architecture candidate with the lowest computational power into accelerators designed for higher computational power requirements. Subsequently, the product of $MC \cdot E \cdot D$ for all accelerators is calculated, and the architecture yielding the minimum product is chosen as the optimal solution (Joint Optimal in Fig. 8).

By comparing the Joint Optimal with the individual Optimal, we find that although the Joint Optimal still has a certain gap (with $MC \cdot E \cdot D$ being on average 34% higher), this gap

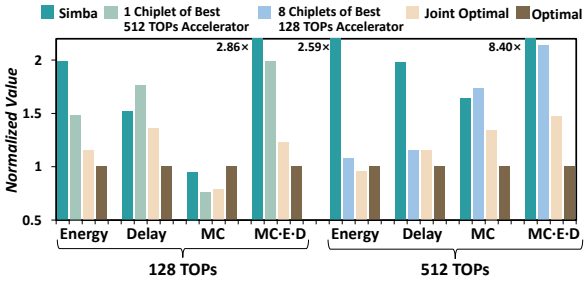

Fig. 8. Energy, Delay, and MC of Four Construction Schemes for 128 TOPs and 512 TOPs Accelerators. These construction schemes are: constructed by Simba chiplets (**Simba**), constructed by the chiplets of the optimal explored architecture of the other computing power (**1 Chiplet of Best 512 TOPs Accelerator** and **8 Chiplets of Best 128 TOPs Accelerator**), constructed by the optimal solution found through joint architecture exploration of the two computing power (**Joint Optimal**), and constructed by the optimal architecture explored under the target of its own computing power (**Optimal**). It is worth noting that the second scheme is feasible because, regrading $MC \cdot E \cdot D$, the optimal architecture for 512 TOPs and 128 TOPs happens to be 4 chiplets and 2 Chiplets (Fig. 6(b)), so one chiplet of 512 TOPs and 8 Chiplets of 128 TOPs can be used to construct a 128 TOPs accelerator and 512 TOPs accelerator, respectively.

is much smaller than those of the previous two approaches. This modest overhead is acceptable when considering that chiplet reuse can significantly reduce NRE costs, including one-time expenses such as design, verification, IP acquisition, and tape-out. The benefits of this approach become increasingly significant in advanced process nodes or fragmented markets since NRE costs tend to grow non-linearly with process advancement, while the fragmented market—with its smaller volumes for each market—makes it challenging to amortize expensive NRE costs effectively. Thus, it concludes that ***with the proper design considerations—facilitated by the support of Gemini—the idea of employing a single chiplet for multiple accelerators can be effectively applied to DNN inference accelerators.***

C. Learn From an Actual SPM Example

In this section, we demonstrate the advantages of Gemini mapping through a practical example (Fig. 9) and analyze how to allocate computing resources to each layer in a multi-core chiplet accelerator.

As can be observed in Fig.9 bottom left, the data volume from $layer_1$ to $layer_2$ and from $layer_2$ to $layer_3$ is significantly higher than that of other dependencies. The links with the most data (reddest) in the Tangram SPM scheme are also caused by these two dependencies. For instance, since the data calculated in $layer_2$ needs to be sent to $layer_3$, under XY-routing, the data calculated by the left cores of $layer_2$ is first sent to the right and then downwards, as illustrated in the upper part of the mesh in Fig.9. The accumulation of ofmaps data computed by multiple cores of $layer_2$ leads to a large amount of communication data on the upper on-chip links. Thus the links are red. During the downward transmission process, each core of $layer_3$ consumes a portion of the ofmaps data calculated by $layer_2$, causing the actual data volume to decrease gradually. This is reflected by the colour of the downward links becoming lighter (except for the D2D links). However, due to the smaller bandwidth of the D2D links compared to the on-chip links, the bandwidth pressure on the D2D links is higher.

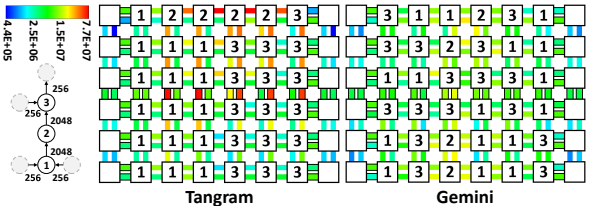

Fig. 9. Network Traffic Heatmap of the Optimal SPM Scheme Explored by Tangram and Gemini on 72 TOPs G-Arch. The nodes without numbers at the edges of the heatmap represent communication nodes within the controllers in the IO Die (Fig. 2(a)). All D2D links in the figure are marked with black borders. Due to the D2D bandwidth being half of the on-chip NoC link, we double the data volume on it to display the bandwidth pressure more clearly. The bottom-left corner depicts the topology and data volume of each dependency of the three-layer workload in Transformer. Layers without numbers represent layers belonging to different layer groups.

By automatically exploring the vast optimization space we have defined, Gemini mapping can discover solutions that are difficult to design manually. In detail, as shown in Gemini scheme in Fig. 9, we can see that the red and orange links have completely disappeared. This is mainly due to two factors: (1) the total hop count decreases by 34.2%, with a 74% reduction in hop count on the intermediate D2D links; and (2) Gemini can utilize the originally relatively idle links (blue) more effectively, which is reflected in the decreased number of blue links in the figure. As a result, the overall network traffic is more evenly distributed, significantly improving network performance and overall performance.

From the actual example in Fig. 9 and other similar instances that are not shown, we can derive an **insight**: ***the widely used clustered core allocation strategy [7], [15], [46], [55], [57], [66], where a layer is assigned to consecutive and rectangle-like core group, may not always be a good choice. This is because the dependencies between certain layers may have particularly high data transfer requirements, and clustering cores for each layer together cannot disperse this transfer demand, leading to network congestion.***

VIII. Conclusion

In this work, we introduce Gemini, a mapping and architecture co-exploration framework for large-scale DNN chiplet accelerators, which considers MC, performance, and energy efficiency. For mapping, Gemini employs a novel encoding method to define LP spatial mapping schemes and the corresponding parsing method. This approach enables us to identify the optimization space of LP SPM and unearth hidden optimization opportunities. Additionally, we devise a specially-tailored SA algorithm to efficiently navigate this space. Regarding architecture, we provide a highly configurable hardware template and precise evaluators for performance, energy, and MC. Our experiments demonstrate that Gemini's explored architecture and mapping scheme significantly outperform the SOTA Simba architecture with Tangram mapping, with only a slight increase in monetary cost. We also present insightful observations about the use of chiplet technology in architecture design and DNN workload mapping within chiplet contexts.

Acknowledgment

This research was partially supported by National Key R&D Program of China (2022YFB2804103), Tsinghua University Dushi Program, and Tsinghua University Talent Program, National Natural Science Foundation of China (62072262).

References

[1] "The proof for comprehensiveness of sa operators," https://anonymous.4open.science/r/2024HPCA_proof-4F15.

[2] "Space calculation of gemini mapping and tangram mapping," https://anonymous.4open.science/r/Space-Calculation-754A/.

[3] M. Alwani, H. Chen, M. Ferdman, and P. A. Milder, "Fused-layer CNN accelerators," in *49th Annual IEEE/ACM International Symposium on Microarchitecture, MICRO 2016, Taipei, Taiwan, October 15-19, 2016*. IEEE Computer Society, 2016, pp. 22:1–22:12. [Online]. Available: https://doi.org/10.1109/MICRO.2016.7783725

[4] AMD, "Epyc-7003," https://www.amd.com/en/processors/epyc-7003-series.

[5] A. Boni, A. Pierazzi, and D. Vecchi, "Lvds i/o interface for gb/s-per-pin operation in 0.35-/spl mu/m cmos," *IEEE Journal of Solid-State Circuits*, vol. 36, no. 4, pp. 706–711, 2001.

[6] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," 2020.

[7] J. Cai, Y. Wei, Z. Wu, S. Peng, and K. Ma, "Inter-layer scheduling space definition and exploration for tiled accelerators," in *Proceedings of the 50th Annual International Symposium on Computer Architecture, ISCA 2023, Orlando, FL, USA, June 17-21, 2023*, Y. Solihin and M. A. Heinrich, Eds. ACM, 2023, pp. 13:1–13:17. [Online]. Available: https://doi.org/10.1145/3579371.3589048

[8] Y. Chen, J. S. Emer, and V. Sze, "Eyeriss: A spatial architecture for energy-efficient dataflow for convolutional neural networks," in *43rd ACM/IEEE Annual International Symposium on Computer Architecture, ISCA 2016, Seoul, South Korea, June 18-22, 2016*. IEEE Computer Society, 2016, pp. 367–379. [Online]. Available: https://doi.org/10.1109/ISCA.2016.40

[9] Y. Chen, J. Li, H. Xiao, X. Jin, S. Yan, and J. Feng, "Dual path networks," in *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett, Eds., 2017, pp. 4467–4475. [Online]. Available: https://proceedings.neurips.cc/paper/2017/hash/f7e0b956540676a129760a3eae309294-Abstract.html

[10] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, J. Burstein, C. Doran, and T. Solorio, Eds. Association for Computational Linguistics, 2019, pp. 4171–4186. [Online]. Available: https://doi.org/10.18653/v1/n19-1423

[11] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," 2021.

[12] dramexchange, "Dram price," https://www.dramexchange.com/.

[13] Y. Feng and K. Ma, "Chiplet actuary: a quantitative cost model and multi-chiplet architecture exploration," in *DAC '22: 59th ACM/IEEE Design Automation Conference, San Francisco, California, USA, July 10 - 14, 2022*, R. Oshana, Ed. ACM, 2022, pp. 121–126. [Online]. Available: https://doi.org/10.1145/3489517.3530428

[14] M. Gao, J. Pu, X. Yang, M. Horowitz, and C. Kozyrakis, "TETRIS: scalable and efficient neural network acceleration with 3d memory," in *Proceedings of the Twenty-Second International Conference on Architectural Support for Programming Languages and Operating Systems, ASPLOS 2017, Xi'an, China, April 8-12, 2017*, Y. Chen, O. Temam, and J. Carter, Eds. ACM, 2017, pp. 751–764. [Online]. Available: https://doi.org/10.1145/3037697.3037702

[15] 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 2019, Providence, RI, USA, April 13-17, 2019*, I. Bahar, M. Herlihy, E. Witchel, and A. R. Lebeck, Eds. ACM, 2019, pp. 807–820. [Online]. Available: https://doi.org/10.1145/3297858.3304014

[16] W. Gomes, A. Koker, P. N. Stover, D. B. Ingerly, S. Siers, S. Venkataraman, C. Pelto, T. Shah, A. Rao, F. O'Mahony, E. Karl, L. Cheney, I. Rajwani, H. Jain, R. Cortez, A. Chandrasekhar, B. Kanthi, and R. Koduri, "Ponte vecchio: A multi-tile 3d stacked processor for exascale computing," in *IEEE International Solid-State Circuits Conference, ISSCC 2022, San Francisco, CA, USA, February 20-26, 2022*. IEEE, 2022, pp. 42–44. [Online]. Available: https://doi.org/10.1109/ISSCC42614.2022.9731673

[17] 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 2016, Las Vegas, NV, USA, June 27-30, 2016*. IEEE Computer Society, 2016, pp. 770–778. [Online]. Available: https://doi.org/10.1109/CVPR.2016.90

[18] K. Hegde, P. Tsai, S. Huang, V. Chandra, A. Parashar, and C. W. Fletcher, "Mind mappings: enabling efficient algorithm-accelerator mapping space search," in *ASPLOS '21: 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Virtual Event, USA, April 19-23, 2021*, T. Sherwood, E. D. Berger, and C. Kozyrakis, Eds. ACM, 2021, pp. 943–958. [Online]. Available: https://doi.org/10.1145/3445814.3446762

[19] P. Huang, C. Lu, W. Wei, C. Chiu, K. Ting, C. Hu, C. Tsai, S. Hou, W. Chiou, C. Wang, and D. Yu, "Wafer level system integration of the fifth generation cowos®-s with high performance si interposer at 2500 mm2," in *2021 IEEE 71st Electronic Components and Technology Conference (ECTC)*. IEEE, 2021, pp. 101–104.

[20] Q. Huang, A. Kalaiah, M. Kang, J. Demmel, G. Dinh, J. Wawrzynek, T. Norell, and Y. S. Shao, "Cosa: Scheduling by constrained optimization for spatial accelerators," in *48th ACM/IEEE Annual International Symposium on Computer Architecture, ISCA 2021, Valencia, Spain, June 14-18, 2021*. IEEE, 2021, pp. 554–566. [Online]. Available: https://doi.org/10.1109/ISCA52012.2021.00050

[21] M. James, M. Tom, P. Groeneveld, and V. Kibardin, "ISPD 2020 physical mapping of neural networks on a wafer-scale deep learning accelerator," in *ISPD 2020: International Symposium on Physical Design, Taipei, Taiwan, March 29 - April 1, 2020, delayed to September 20-23, 2020*, W. Swartz and J. Lienig, Eds. ACM, 2020, pp. 145–149. [Online]. Available: https://doi.org/10.1145/3372780.3380846

[22] Z. Jia, S. Lin, C. R. Qi, and A. Aiken, "Exploring hidden dimensions in parallelizing convolutional neural networks," in *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, ser. Proceedings of Machine Learning Research, J. G. Dy and A. Krause, Eds., vol. 80. PMLR, 2018, pp. 2279–2288. [Online]. Available: http://proceedings.mlr.press/v80/jia18a.html

[23] Y. Jiao, L. Han, R. Jin, Y. Su, C. Ho, L. Yin, Y. Li, L. Chen, Z. Chen, L. Liu, Z. He, Y. Yan, J. He, J. Mao, X. Zai, X. Wu, Y. Zhou, M. Gu, G. Zhu, R. Zhong, W. Lee, P. Chen, Y. Chen, W. Li, D. Xiao, Q. Yan, M. Zhuang, J. Chen, Y. Tian, Y. Lin, W. Wu, H. Li, and Z. Dou, "A 12nm programmable convolution-efficient neural-processing-unit chip achieving 825tops," in *2020 IEEE International Solid- State Circuits Conference, ISSCC 2020, San Francisco, CA, USA, February 16-20, 2020*. IEEE, 2020, pp. 136–140. [Online]. Available: https://doi.org/10.1109/ISSCC19947.2020.9062984

[24] N. P. Jouppi, D. H. Yoon, M. Ashcraft, M. Gottscho, T. B. Jablin, G. Kurian, J. Laudon, S. Li, P. C. Ma, X. Ma, T. Norrie, N. Patil, S. Prasad, C. Young, Z. Zhou, and D. A. Patterson, "Ten lessons from three generations shaped google's tpuv4i : Industrial product," in *48th ACM/IEEE Annual International Symposium on Computer Architecture, ISCA 2021, Valencia, Spain, June 14-18, 2021*. IEEE, 2021, pp. 1–14. [Online]. Available: https://doi.org/10.1109/ISCA52012.2021.00010

[25] S. Kao, G. Jeong, and T. Krishna, "Confuciux: Autonomous hardware resource assignment for DNN accelerators using reinforcement learning," in *53rd Annual IEEE/ACM International Symposium on Microarchitecture, MICRO 2020, Athens, Greece, October 17-21, 2020*. IEEE, 2020, pp. 622–636. [Online]. Available: https://doi.org/10.1109/MICRO50266.2020.00058

[26] S. Kao and T. Krishna, "GAMMA: automating the HW mapping of DNN models on accelerators via genetic algorithm," in *IEEE/ACM International Conference On Computer Aided Design, ICCAD 2020, San Diego, CA, USA, November 2-5, 2020*. IEEE, 2020, pp. 44:1–44:9. [Online]. Available: https://doi.org/10.1145/3400302.3415639

[27] S. Kirkpatrick, C. D. Gelatt Jr, and M. P. Vecchi, "Optimization by simulated annealing," *science*, vol. 220, no. 4598, pp. 671–680, 1983.

[28] S. Knowles, "Graphcore," in *IEEE Hot Chips 33 Symposium, HCS 2021, Palo Alto, CA, USA, August 22-24, 2021*. IEEE, 2021, pp. 1–25. [Online]. Available: https://doi.org/10.1109/HCS52781.2021.9567075

[29] H. Kwon, P. Chatarasi, M. Pellauer, A. Parashar, V. Sarkar, and T. Krishna, "Understanding reuse, performance, and hardware cost of DNN dataflow: A data-centric approach," in *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture, MICRO 2019, Columbus, OH, USA, October 12-16, 2019*. ACM, 2019, pp. 754–768. [Online]. Available: https://doi.org/10.1145/3352460.3358252

[30] H. Kwon, A. Samajdar, and T. Krishna, "MAERI: enabling flexible dataflow mapping over DNN accelerators via reconfigurable

interconnects," in *Proceedings of the Twenty-Third International Conference on Architectural Support for Programming Languages and Operating Systems, ASPLOS 2018, Williamsburg, VA, USA, March 24-28, 2018*, X. Shen, J. Tuck, R. Bianchini, and V. Sarkar, Eds. ACM, 2018, pp. 461–475. [Online]. Available: https://doi.org/10.1145/3173162.3173176

[31] H. Liao, J. Tu, J. Xia, and X. Zhou, "Davinci: A scalable architecture for neural network computing," in *2019 IEEE Hot Chips 31 Symposium (HCS), Cupertino, CA, USA, August 18-20, 2019*. IEEE, 2019, pp. 1–44. [Online]. Available: https://doi.org/10.1109/HOTCHIPS.2019.8875654

[32] C. Liu, B. Zoph, M. Neumann, J. Shlens, W. Hua, L. Li, L. Fei-Fei, A. L. Yuille, J. Huang, and K. Murphy, "Progressive neural architecture search," in *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part I*, ser. Lecture Notes in Computer Science, V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, Eds., vol. 11205. Springer, 2018, pp. 19–35. [Online]. Available: https://doi.org/10.1007/978-3-030-01246-5_2

[33] L. Lu, N. Guan, Y. Wang, L. Jia, Z. Luo, J. Yin, J. Cong, and Y. Liang, "TENET: A framework for modeling tensor dataflow based on relation-centric notation," in *48th ACM/IEEE Annual International Symposium on Computer Architecture, ISCA 2021, Valencia, Spain, June 14-18, 2021*. IEEE, 2021, pp. 720–733. [Online]. Available: https://doi.org/10.1109/ISCA52012.2021.00062

[34] X. Ma, C. Si, Y. Wang, C. Liu, and L. Zhang, "NASA: accelerating neural network design with a NAS processor," in *48th ACM/IEEE Annual International Symposium on Computer Architecture, ISCA 2021, Valencia, Spain, June 14-18, 2021*. IEEE, 2021, pp. 790–803. [Online]. Available: https://doi.org/10.1109/ISCA52012.2021.00067

[35] G. Moore, "Cramming more components onto integrated circuits," *Electronics*, vol. 38, 04 1965.

[36] S. Naffziger, N. Beck, T. Burd, K. Lepak, G. H. Loh, M. Subramony, and S. White, "Pioneering chiplet technology and design for the amd epyc™ and ryzen™ processor families : Industrial product," in *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*, 2021, pp. 57–70.

[37] S. Naffziger, K. Lepak, M. Paraschou, and M. Subramony, "2.2 AMD chiplet architecture for high-performance server and desktop products," in *2020 IEEE International Solid- State Circuits Conference, ISSCC 2020, San Francisco, CA, USA, February 16-20, 2020*. IEEE, 2020, pp. 44–45. [Online]. Available: https://doi.org/10.1109/ISSCC19947.2020.9063103

[38] T. Norrie, N. Patil, D. H. Yoon, G. Kurian, S. Li, J. Laudon, C. Young, N. P. Jouppi, and D. A. Patterson, "Google's training chips revealed: Tpuv2 and tpuv3," in *IEEE Hot Chips 32 Symposium, HCS 2020, Palo Alto, CA, USA, August 16-18, 2020*. IEEE, 2020, pp. 1–70. [Online]. Available: https://doi.org/10.1109/HCS49909.2020.9220735

[39] Nvidia, "Nvdla deep learning accelerator," http://nvdla.org., 2017.

[40] OpenAI, "Gpt-4 technical report," 2023.

[41] A. Parashar, P. Raina, Y. S. Shao, Y. Chen, V. A. Ying, A. Mukkara, R. Venkatesan, B. Khailany, S. W. Keckler, and J. S. Emer, "Timeloop: A systematic approach to DNN accelerator evaluation," in *IEEE International Symposium on Performance Analysis of Systems and Software, ISPASS 2019, Madison, WI, USA, March 24-26, 2019*. IEEE, 2019, pp. 304–315. [Online]. Available: https://doi.org/10.1109/ISPASS.2019.00042

[42] J. W. Poulton, W. J. Dally, X. Chen, J. G. Eyles, T. H. Greer, S. G. Tell, J. M. Wilson, and C. T. Gray, "A 0.54 pj/b 20 gb/s ground-referenced single-ended short-reach serial link in 28 nm cmos for advanced packaging applications," *IEEE Journal of Solid-State Circuits*, vol. 48, no. 12, pp. 3206–3218, 2013.

[43] 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, 2019.

[44] V. J. Reddi, C. Cheng, D. Kanter, P. Mattson, G. Schmuelling, C. Wu, B. Anderson, M. Breughe, M. Charlebois, W. Chou, R. Chukka, C. Coleman, S. Davis, P. Deng, G. Diamos, J. Duke, D. Fick, J. S. Gardner, I. Hubara, S. Idgunji, T. B. Jablin, J. Jiao, T. S. John, P. Kanwar, D. Lee, J. Liao, A. Lokhmotov, F. Massa, P. Meng, P. Micikevicius, C. Osborne, G. Pekhimenko, A. T. R. Rajan, D. Sequeira, A. Sirasao, F. Sun, H. Tang, M. Thomson, F. Wei, E. Wu, L. Xu, K. Yamada, B. Yu, G. Yuan, A. Zhong, P. Zhang, and Y. Zhou, "Mlperf inference benchmark," in *47th ACM/IEEE Annual International Symposium on Computer Architecture, ISCA 2020, Valencia, Spain, May 30 - June 3, 2020*. IEEE, 2020, pp. 446–459. [Online]. Available: https://doi.org/10.1109/ISCA45697.2020.00045

[45] A. Shafiee, A. Nag, N. Muralimanohar, R. Balasubramonian, J. P. Strachan, M. Hu, R. S. Williams, and V. Srikumar, "ISAAC: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars," in *43rd ACM/IEEE Annual International Symposium on Computer Architecture, ISCA 2016, Seoul, South Korea, June 18-22, 2016*. IEEE Computer Society, 2016, pp. 14–26. [Online]. Available: https://doi.org/10.1109/ISCA.2016.12

[46] 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 *Proceedi .ngs of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO '52. New York, NY, USA: Association for Computing Machinery, 2019, p. 14–27. [Online]. Available: https://doi.org/10.1145/3352460.3358302

[47] R. Shivnaraine, M. van Ierssel, K. Farzan, D. DiClemente, G. Ng, N. Wang, J. Musayev, G. Dutta, M. Shibata, A. Moradi, H. Vahedi, M. Farzad, P. Kainth, M. Yu, N. Nguyen, J. Pham, and A. McLaren, "11.2 A 26.5625-to-106.25gb/s XSR serdes with 1.55pj/b efficiency in 7nm CMOS," in *IEEE International Solid-State Circuits Conference, ISSCC 2021, San Francisco, CA, USA, February 13-22, 2021*. IEEE, 2021, pp. 181–183. [Online]. Available: https://doi.org/10.1109/ISSCC42613.2021.9365975

[48] A. Shokrollahi, D. A. Carnelli, J. Fox, K. L. Hofstra, B. Holden, A. Hormati, P. Hunt, M. Johnston, J. Keay, S. Pesenti, R. Simpson, D. Stauffer, A. Stewart, G. Surace, A. Tajalli, O. T. Amiri, A. Tschank, R. Ulrich, C. Walter, F. Licciardello, Y. Mogentale, and A. Singh, "10.1 A pin-efficient 20.83gb/s/wire 0.94pj/bit forwarded clock cnrz-5-coded serdes up to 12mm for MCM packages in 28nm CMOS," in *2016 IEEE International Solid-State Circuits Conference, ISSCC 2016, San Francisco, CA, USA, January 31 - February 4, 2016*. IEEE, 2016, pp. 182–183. [Online]. Available: https://doi.org/10.1109/ISSCC.2016.7417967

[49] A. Singh, D. A. Carnelli, A. Falay, K. L. Hofstra, F. Licciardello, K. Salimi, H. Santos, A. Shokrollahi, R. Ulrich, C. Walter, J. Fox, P. Hunt, J. Keay, R. Simpson, A. Stewart, G. Surace, and H. S. Cronie, "26.3 A pin- and power-efficient low-latency 8-to-12gb/s/wire 8b8w-coded serdes link for high-loss channels in 40nm technology," in *2014 IEEE International Conference on Solid-State Circuits Conference, ISSCC 2014, Digest of Technical Papers, San Francisco, CA, USA, February 9-13, 2014*. IEEE, 2014, pp. 442–443. [Online]. Available: https://doi.org/10.1109/ISSCC.2014.6757505

[50] D. C. Stow, I. Akgun, R. Barnes, P. Gu, and Y. Xie, "Cost analysis and cost-driven IP reuse methodology for soc design based on 2.5d/3d integration," in *Proceedings of the 35th International Conference on Computer-Aided Design, ICCAD 2016, Austin, TX, USA, November 7-10, 2016*, F. Liu, Ed. ACM, 2016, p. 56. [Online]. Available: https://doi.org/10.1145/2966986.2980095

[51] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," in *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA*, S. P. Singh and S. Markovitch, Eds. AAAI Press, 2017, pp. 4278–4284. [Online]. Available: http://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14806

[52] E. Talpes, D. Williams, and D. D. Sarma, "DOJO: the microarchitecture of tesla's exa-scale computer," in *2022 IEEE Hot Chips 34 Symposium, HCS 2022, Cupertino, CA, USA, August 21-23, 2022*. IEEE, 2022, pp. 1–28. [Online]. Available: https://doi.org/10.1109/HCS55958.2022.9895534

[53] Z. Tan, H. Cai, R. Dong, and K. Ma, "Nn-baton: DNN workload orchestration and chiplet granularity exploration for multichip accelerators," in *48th ACM/IEEE Annual International Symposium on Computer Architecture, ISCA 2021, Valencia, Spain, June 14-18, 2021*. IEEE, 2021, pp. 1013–1026. [Online]. Available: https://doi.org/10.1109/ISCA52012.2021.00083

[54] UCIe, "Ucie1.1 specifications," https://www.uciexpress.org/specifications.

[55] J. Vasiljevic, L. Bajic, D. Capalija, S. Sokorac, D. Ignjatovic, L. Bajic, M. Trajkovic, I. Hamer, I. Matosevic, A. Cejkov, U. Aydonat, T. Zhou, S. Z. Gilani, A. Paiva, J. Chu, D. Maksimovic, S. A. Chin, Z. Moudallal, A. Rakhmati, S. Nijjar, A. Bhullar, B. Drazic, C. Lee, J. Sun, K. Kwong, J. Connolly, M. Dooley, H. Farooq, J. Y. T. Chen, M. Walker, K. Dabiri, K. Mabee, R. S. Lal, N. Rajatheva, R. Retnamma, S. Karodi, D. Rosen, E. Munoz, A. Lewycky, A. Knezevic, R. Kim, A. Rui, A. Drouillard, and D. Thompson, "Compute substrate for software 2.0," *IEEE Micro*, vol. 41, no. 2, pp. 50–55, 2021. [Online]. Available: https://doi.org/10.1109/MM.2021.3061912

[56] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS'17. Red Hook, NY, USA: Curran Associates Inc., 2017, p. 6000–6010.

[57] S. Venkataramani, A. Ranjan, S. Banerjee, D. Das, S. Avancha, A. Jagannathan, A. Durg, D. Nagaraj, B. Kaul, P. Dubey, and A. Raghunathan, "Scaledeep: A scalable compute architecture for learning and evaluating deep networks," in *Proceedings of the 44th Annual International Symposium on Computer Architecture, ISCA 2017, Toronto, ON, Canada, June 24-28, 2017*. ACM, 2017, pp. 13–26. [Online]. Available: https://doi.org/10.1145/3079856.3080244

[58] R. Venkatesan, Y. S. Shao, M. Wang, J. Clemons, S. Dai, M. Fojtik, B. Keller, A. Klinefelter, N. R. Pinckney, P. Raina, Y. Zhang, B. Zimmer, W. J. Dally, J. S. Emer, S. W. Keckler, and B. Khailany, "Magnet: A modular accelerator generator for neural networks," in *Proceedings of the International Conference on Computer-Aided Design, ICCAD 2019, Westminster, CO, USA, November 4-7, 2019*, D. Z. Pan, Ed. ACM, 2019, pp. 1–8. [Online]. Available: https://doi.org/10.1109/ICCAD45719.2019.8942127

[59] P. Vivet, E. Guthmuller, Y. Thonnart, G. Pillonnet, G. Moritz, I. Miro-Panadès, C. F. Tortolero, J. Durupt, C. Bernard, D. Varreau, J. J. H. Pontes, S. Thuries, D. Coriat, M. Harrand, D. Dutoit, D. Lattard, L. Arnaud, J. Charbonnier, P. Coudrain, A. Garnier, F. Berger, A. Gueugnot, A. Greiner, Q. L. Meunier, A. Farcy, A. Arriordaz, S. Cheramy, and F. Clermidy, "A 220gops 96-core processor with 6 chiplets 3d-stacked on an active interposer offering 0.6ns/mm latency, 3tb/s/mm^2 inter-chiplet interconnects and 156mw/mm^2@ 82%-peak-efficiency DC-DC converters," in *2020 IEEE International Solid-State Circuits Conference, ISSCC 2020, San Francisco, CA, USA, February 16-20, 2020*. IEEE, 2020, pp. 46–48. [Online]. Available: https://doi.org/10.1109/ISSCC19947.2020.9062927

[60] H. Wang, X. Zhu, L. Peh, and S. Malik, "Orion: a power-performance simulator for interconnection networks," in *Proceedings of the 35th Annual International Symposium on Microarchitecture, Istanbul, Turkey, November 18-22, 2002*, E. R. Altman, K. Ebcioglu, S. A. Mahlke, B. R. Rau, and S. J. Patel, Eds. ACM/IEEE Computer Society, 2002, pp. 294–305. [Online]. Available: https://doi.org/10.1109/MICRO.2002.1176258

[61] O. Wechsler, M. Behar, and B. Daga, "Spring hill (NNP-I 1000) intel's data center inference chip," in *2019 IEEE Hot Chips 31 Symposium (HCS), Cupertino, CA, USA, August 18-20, 2019*. IEEE, 2019, pp. 1–12. [Online]. Available: https://doi.org/10.1109/HOTCHIPS.2019.8875671

[62] Q. Xiao, S. Zheng, B. Wu, P. Xu, X. Qian, and Y. Liang, "HASCO: towards agile hardware and software co-design for tensor computation," in *48th ACM/IEEE Annual International Symposium on Computer Architecture, ISCA 2021, Valencia, Spain, June 14-18, 2021*. IEEE, 2021, pp. 1055–1068. [Online]. Available: https://doi.org/10.1109/ISCA52012.2021.00086

[63] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 5987–5995.

[64] A. Yang, "Deep learning training at scale spring crest deep learning accelerator (intel® nervana™ NNP-T)," in *2019 IEEE Hot Chips 31 Symposium (HCS), Cupertino, CA, USA, August 18-20, 2019*. IEEE, 2019, pp. 1–20. [Online]. Available: https://doi.org/10.1109/HOTCHIPS.2019.8875643

[65] X. Yang, M. Gao, Q. Liu, J. Setter, J. Pu, A. Nayak, S. Bell, K. Cao, H. Ha, P. Raina, C. Kozyrakis, and M. Horowitz, "Interstellar: Using halide's scheduling language to analyze DNN accelerators," in *ASPLOS '20: Architectural Support for Programming Languages and Operating Systems, Lausanne, Switzerland, March 16-20, 2020*, J. R. Larus, L. Ceze, and K. Strauss, Eds. ACM, 2020, pp. 369–383. [Online]. Available: https://doi.org/10.1145/3373376.3378514

[66] S. Zheng, X. Zhang, L. Liu, S. Wei, and S. Yin, "Atomic dataflow based graph-level workload orchestration for scalable DNN accelerators," in *IEEE International Symposium on High-Performance Computer Architecture, HPCA 2022, Seoul, South Korea, April 2-6, 2022*. IEEE, 2022, pp. 475–489. [Online]. Available: https://doi.org/10.1109/HPCA53966.2022.00042

[67] S. Zheng, X. Zhang, D. Ou, S. Tang, L. Liu, S. Wei, and S. Yin, "Efficient scheduling of irregular network structures on cnn accelerators," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 11, pp. 3408–3419, 2020.

[68] B. Zimmer, R. Venkatesan, Y. S. Shao, J. Clemons, M. Fojtik, N. Jiang, B. Keller, A. Klinefelter, N. R. 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 J. Solid State Circuits*, vol. 55, no. 4, pp. 920–932, 2020. [Online]. Available: https://doi.org/10.1109/JSSC.2019.2960488

APPENDIX

A. Abstract

This appendix provides guidance on accessing and using the Gemini framework (introduced in Sec. V) to replicate the key results shown in Fig. 5. The process is divided into two steps: (1) Conducting DSE for a 72 TOPs setup (Table. I) to identify the optimal architecture, referred to as G-ARCH; (2) Comparing the efficacy of G-ARCH with G-MAP against the baseline architecture S-ARCH and baseline mapping T-MAP, across varying batch sizes (introduced in Sec. VI-A) and five different networks (introduced in Sec. VI-A3). The remaining experiments, which also involve similar DSE and analysis, are omitted here for the sake of brevity.

B. Artifact check-list (meta-information)

- **Algorithm: Simulated Annealing, Exhaustive Search and Dynamic Programming**
- **Program: C++, Shell, Python (only for data collection)**
- **Compilation: by Makefile**
- **Hardware: It is best to have a server with more than 80 threads.**
- **Metrics: Cost function ($MC^{\alpha} \times E^{\beta} \times D^{\gamma}$, $MC \times E \times D$ is employed in DSE. $E \times D$ is employed in the comparisons with baseline.)**
- **Experiments: reproduce Fig. 5, including 72TOPs DSE and comparisons with baseline.**
- **How much disk space required (approximately)?: 1GB**
- **How much time is needed to prepare workflow(approximately)?: Several minutes at most.**
- **How much time is needed to complete experiments (approximately)?: For DSE, it takes about 38 minutes with 80 threads; For comparison, it takes about 14 minutes using 10 threads. The server is Intel Xeon Platinum 8260 server**
- **Publicly available?: Yes**
- **Code licenses (if publicly available)?: BSD 3-Clause License**
- **Archived (provide DOI)?: 10.5281/zenodo.10207613**

C. Description

1) How to access: The artifact is uploaded to Zenodo:10.5281/zenodo.10207613

2) Software dependencies: A C++ compilation environment with support for the C++ 17 standard is required. Linux is recommended. It is recommended to use "GNU make" to build the program. Additionally, two python packages "pandas" and "csv" are needed.

D. Installation

For artifact evaluation, start by downloading the artifact from Zenodo:

```
$ wget -O GEMINI_AE.zip https://zenodo.org/records
    /10207613/files/GEMINI_AE.zip?download=1
$ unzip GEMINI_AE.zip
```

Our Gemini exploration framework is in "GEMINI". We use Makefile to build the GEMINI framework.

```
$ cd GEMINI
$ make
```

Then the executable target will be generated at "./build/stschedule".

You can install the needed python packages using pip with following commands.

```
$ pip install -r requirements.txt
```

Or you can use conda to install with following commands.

```
$ conda install --file requirements.txt
```

E. Experiment workflow

1) 72TOPs DSE: Once GEMINI framework is built, you can execute DSE experiment and search the best arch with command below.

```
$ ./dse.sh
```

The DSE script uses 80 threads and takes around 38 minutes to run on Intel Xeon Platinum 8260 server. Then you can see the state of current running DSE. Once the process is completed, the command-line window will output the optimal architecture. You can compare this optimal architecture with the one mentioned in Sec. VI-B, which is (2, 36, 144GB/s, 32GB/s, 16GB/s, 2MB, 1024). In the "dse_log" folder, there will be a folder named with the timestamp, containing the outputs for each architecture candidate, as well as the summarized "result.csv" file. You can compare the generated "result.csv" with the "DSE_result.csv" provided in "expected_results" folder, which is employed in our paper.

2) Comparison with Baselines: Once you obtain the optimal architecture, you can proceed with the comparison of G-ARCH+G-MAP, S-ARCH+T-MAP, and S-ARCH+G-MAP for the five networks and two batch sizes to reproduce Fig. 5 with following commands.

```
$ ./compare.sh <output_dir>/best_arch.txt
```

Please note that "<output_dir>" refers to the address where the "best_arch.txt" file is stored, which is the folder named with a timestamp within the "DSE_log" directory like ./dse_log/2023_11_08_10_56_52/best_arch.txt.

The script uses 10 threads and takes around 14 minutes to run on Intel Xeon Platinum 8260 server. Then you can see the output information in the terminal window about the optimization rate of G-ARCH+G-MAP over S-ARCH+T-MAP, which will be identical with the results posted in Abstract and Sec. VI-B(1.98× performance improvement, 1.41× energy efficiency improvement, and 14.3% MC increase).

Then you can run the following command to reproduce the Excel data of Fig. 5. The Fig_5.csv file, generated in the current directory, can be validated through comparison with the Fig_5.csv file, which contains the data used in Fig. 5, located in the expected_results folder.

```
$ python3 Fig5_reproduce.py <output_dir>/compare.
    csv
```

Please note that "<output_dir>" refers to the address where the "compare.csv" file is stored, which is the folder named with a timestamp within the 'Compare_log' directory like ./compare_log/2023_11_08_10_56_52/compare.csv.

F. Evaluation and expected results

There are two key data results: the optimal architecture discovered by DSE (2, 36, 144GB/s, 32GB/s, 16GB/s, 2MB, 1024) and the improvement ratio of G-ARCH+G-MAP compared to S-ARCH+T-MAP (1.98× performance improvement, 1.41× energy efficiency improvement, and 14.3% MC increase). The optimal architecture and improvement ratio will be displayed in the windows after executing dse.sh and compare.sh scripts respectively, and can be verified against the mentioned results. Moreover, the breakdown of energy consumption and delay data for Fig. 5 can be verified by comparing the "Fig_5.csv" file generated in GEMINI folder with the "Fig_5.csv" located in the "expected_results" directory.

G. Methodology

Submission, reviewing and badging methodology:

- https://www.acm.org/publications/policies/artifact-review-badging
- http://cTuning.org/ae/submission-20201122.html
- http://cTuning.org/ae/reviewing-20201122.html