2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA) | 978-1-6654-3333-4/21/$31.00 ©2021 IEEE | DOI: 10.1109/ISCA52012.2021.00083

NN-Baton: DNN Workload Orchestration and Chiplet Granularity Exploration for Multichip Accelerators

Zhanhong Tan*, Hongyu Cai*, Runpei Dong†, Kaisheng Ma*
*Tsinghua University, †Xi'an Jiaotong University
tanzh19@mails.tsinghua.edu.cn, h.tsai@hotmail.com,
runpei.dong@gmail.com, kaisheng@mail.tsinghua.edu.cn

***Abstract*—The revolution of machine learning poses an unprecedented demand for computation resources, urging more transistors on a single monolithic chip, which is not sustainable in the Post-Moore era. The multichip integration with small functional dies, called chiplets, can reduce the manufacturing cost, improve the fabrication yield, and achieve die-level reuse for different system scales. DNN workload mapping and hardware design space exploration on such multichip systems are critical, but missing in the current stage.**

This work provides a hierarchical and analytical framework to describe the DNN mapping on a multichip accelerator and analyze the communication overhead. Based on this framework, we propose an automatic tool called NN-Baton with a *pre-design flow* and a *post-design flow*. The *pre-design flow* aims to guide the chiplet granularity exploration with given area and performance budgets for the target workload. The *post-design flow* focuses on the workload orchestration on different computation levels - package, chiplet, and core - in the hierarchy. Compared to Simba, NN-Baton generates mapping strategies that save 22.5%∼44% energy under the same computation and memory configurations. The architecture exploration demonstrates that area is a decisive factor for the chiplet granularity. For a 2048-MAC system under a 2 mm^2 chiplet area constraint, the 4-chiplet implementation with 4 cores and 16 lanes of 8-size vector-MAC is always the top-pick computation allocation across several benchmarks. In contrast, the optimal memory allocation policy in the hierarchy typically depends on the neural network models.

***Index Terms*—Accelerator, chiplet, MCM, neural network, deep learning, scheduling, design space exploration**

I. Introduction

Applications across the cloud and edge platforms are booming with the flourishing of deep neural networks (DNNs), such as image recognition [20, 31, 57], object detection [16, 52], image caption [12, 63], and natural language processing [21, 29]. Given that state-of-the-art DNN models feature intensive computation and storage [69], a significant number of DNN accelerators have been recently proposed from academia to industry, including FPGAs [9, 65, 73], ASIC accelerators [26, 30, 45, 48], and SoCs in the mobile device and datacenter infrastructure [37, 38, 59].

The area of DNN hardware tends to be increasingly larger [3, 24, 25, 46] to provide high computing throughput and on-chip memory capability. However, facing the end of Moore's Law [33], transistor cost reduction slows and the fabrication yield challenge grows. Consequently, designing such a large monolithic chip leads to pricey fabrication and prolonged development timelines [34].

In the past decades, overcoming the well-known "power wall" by multi-core processors [43] has achieved orders-of-magnitude efficiency improvements. Moreover, to conquer the "memory wall" [67], the novel Non-von Neumann architecture is a promising solution and has been actively studied [54, 70, 75]. Nowadays, a new wall, namely "area wall", has been established, which means that we fail to obtain high integration via a large chip cost-efficiently due to the decline of fabrication yield and the increase of cost per transistor in the advanced process. This challenge motivates us to employ the multi-chip-module (MCM) package with several small functional dies, called chiplets, to provide a large-scale system, including CPUs [4, 58, 64], GPUs [2, 72], FPGAs [18] and DNN accelerators [23, 55, 78].

On-package integration enjoys lower manufacturing costs, higher fabrication yield, and better flexibility than the monolithic chip. Nevertheless, this novel platform for DNN accelerators raises several brand-new challenges for the computer architecture community. Simba [55] is a pioneering multichip DNN accelerator with up to 36 chiplets on the package, and its main contribution is the solution to the natural non-uniformity between on-chip and on-package bandwidth and latency. Beyond that, we clarify the following two challenges.

The first challenge is to map the DNN workload on the multichip accelerator during the hardware's deployment. Recent works concerning DNN mapping mostly focus on a single chip [15, 32, 71] rather than an on-package system with DRAM and multiple accelerator chiplets. According to Simba's prototype, a multichip accelerator involves more complex communication, especially the newly added die-to-die communication, which consumes higher energy and provides lower bandwidth than the on-chip interconnection [66]. Therefore, the neural network workload for chiplets needs to be aggregated as locally as possible. Fragmented data are likely to bring redundant transfer and communication congestion. The multichip system also shows hierarchical parallelism with thousands of processing elements. As a result, the workload mapping requires to be layered and ruled.

The second challenge is to explore the chiplet granularity during the development of an accelerator. Given a required

978-1-6654-3333-4/21/$31.00 ©2021 IEEE
DOI 10.1109/ISCA52012.2021.00083

performance, we expect to select the appropriate chiplet scale and resource allocation policy: the number of MAC (Multiply-and-Accumulate) units and on-chip memory sizes of each level in the hierarchy. A design with fewer chiplets leads to a larger area but simplifies the NoP (Network-on-Package) topology with less inter-chip communication overhead. On the contrary, a large-scale system with abundant chiplets in small sizes (e.g., the 6×6 Simba system) results in complex networking but saves the development cost (e.g., only 6mm^2 each Simba chiplet). An automatic tool for the design space exploration appropriate for the multichip accelerator is urgently in need.

To address the challenges outlined above, this paper aims to (1) describe and analyze the DNN mapping on a multichip accelerator and (2) explore the design space for the optimal chiplet granularity.

Therefore, we first establish a universal and concise hardware model with three levels: package, chiplet, and core that are available for the subsequent analysis. Each hierarchy includes the corresponding computation and memory components. This paper targets the layer-wise mapping on the general-scale multichip accelerator, so we employ the directional ring network on package interconnecting 1-to-8 chiplets rather than an intricate network for tens of chiplets.

Next, we propose an output-centric hierarchical description with spatial and temporal primitives. The spatial primitive is used to partition a large workload to several parallel modules, such as chiplets and cores. In contrast, the temporal primitive is adopted to depict the loop unrolling strategy, aiming to maximize the buffer locality and the data reuse efficiency. Based on the description, we put forward a quantitative and analytical methodology named C^{3}P (Critical-Capacity Critical-Position) to evaluate the hardware and generate the optimal mapping strategy for each layer.

Finally, we propose an automatic tool named NN-Baton. It provides a pre-design flow to guide the chiplet granularity and conduct the multichip accelerator implementation, including the computation and memory resource allocation on each parallel level (package, chiplet, and core). It also contains a post-design flow with the spatial and temporal partition strategies to map each layer on the accelerator.

In summary, this paper makes the following contributions:

- **A Universal Multichip Accelerator:** It consists of three parallel hierarchies: package, chiplet and core to enable the efficient workload mapping and design exploration.
- **A Hierarchical and Analytical Framework:** The framework contains a chiplet-friendly output-centric description with temporal and spatial primitives and a C^{3}P methodology to evaluate the communication overhead.
- **An Automatic Tool:** Based on the output-centric description and the C^{3}P methodology, we develop NN-Baton, a promising automatic tool to guide the development of multichip accelerator and the DNN workload deployment.
- We demonstrate the effectiveness of NN-Baton using several typical models. NN-Baton saves 22.5%∼44% energy compared to Simba in a 4-chiplet system. Additionally, we present how NN-Baton effectually provides the optimal accelerators with the given number of MAC units and area constraint for the target models.

To the best of our knowledge, this is the first work to propose a complete framework with an automatic tool to facilitate the DNN workload mapping and the chiplet granularity exploration for multichip accelerators.

II. Background

This section aims to present the basic core concepts of the DNN workload and chiplet technique, which helps readers to understand the motivations and principles of this work.

A. DNN Workload

DNNs are composed of various components, including convolution, activation function, batch normalization, pooling, and fully-connected layers. The convolution layer, used for feature extraction, is particularly computation-intensive. Figure 1 provides an overview of the convolutional computation with a seven-dimensional loop nest. This paper focuses on the batch size of one to minimize the inference latency in some deployment scenarios like self-driving and datacenter [19, 27, 35]. We define a complete output cube of HO×WO×CO as a layer workload consuming a 3D input cube and a 4D weight tensor. Generally, due to the on-chip resource limitation, a DNN accelerator needs to execute numerous iterations to form the whole workload. When the convolution stride is smaller than the kernel size, calculating adjacent planar convolution windows generates an input overlap (or called halo) region, potentially leading to redundant memory access.

In order to reuse data in the on-chip buffer, DNN accelerators require proper loop tiling and loop unrolling that are critical optimization goals in many recent works [6, 15, 32, 41, 50, 71, 76]. Loop tiling is used to partition a large loop into several tiny ones aiming to fit the buffer size and exploit the memory locality. Loop unrolling targets the order in the loop nest and it decides the dataflow in a computation array, such as input stationary (IS) [50], weight stationary (WS) [47], and output-stationary (OS) [10, 44].

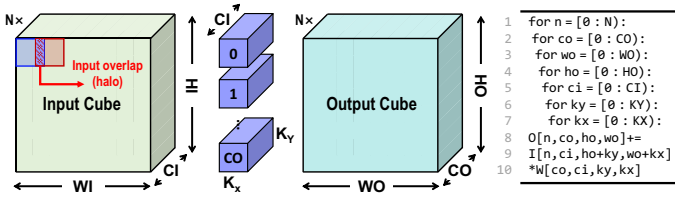


Fig. 1: Overview of a convolution layer.

B. Chiplet Basis

The revolution of big data and machine learning is posing an unprecedented demand for computation resources, which requires more and more transistors to be integrated on the chip. However, due to the slowdown of Dennard scaling and Moore's law [11], it becomes challenging and cost-inefficient to design such a sophisticated SoC with ever-growing area requirement. Tesla FSD with 72 TOPS performance targets at

the self-driving scenario and requires a 260 mm^2 area [3]. The area of Hanguang is as large as 709 mm^2 fabricated in a 12 nm process technology to provide 825 TOPS performance in datacenter [25]. The larger design consumes higher manufacturing costs and leads to a lower fabrication yield.

Chiplet technique becomes one of the promising methods to tackle this problem. Several small functional dies known as chiplet can be assembled to build a large-scale system within a single package using the MCM technique. Chiplets are connected via on-package links in a silicon interposer [39, 64] or an organic substrate [58, 78]. Compared to the reticle-limited large monolithic die, the chiplet-based system enjoys a lower manufacturing & testing cost and a higher fabrication yield [4]. In addition, the chiplet-based system can mix silicon dies from several heterogeneous process nodes to improve cost efficiency [58]. However, the on-package die-to-die communication induces a considerable gap with the traditional fully on-chip implementation and has become an active research topic in recent years [40, 56, 66]. These works target at bridging the gap of bandwidth and energy between the die-to-die and on-chip interconnection.

In the field of DNN accelerators, Simba [55, 78, 79] is a pioneering work fabricated in a chiplet-based system. Unlike previous works using a small-scale 2D parallel array [1, 5, 7, 10, 28, 36, 50], a multichip DNN accelerator contains several computation and memory levels, leading to the complicated workload mapping and energy breakdown. A multichip accelerator tends to provide a large-scale system with multiple arrays in a chiplet and multiple chiplets in a package. Table I shows an overview of the energy breakdown in a multichip accelerator. In contrast with small-scale designs [7, 44, 74], a large L2 memory is introduced to cache data for several arrays. The die-to-die communication has become a significant energy portion, ranking only the second to the DRAM access. In practice, each 1-bit transfer consumes 3.34 pJ energy because the data need to go through a pair of die-to-die PHYs.

Therefore, when we map a workload to several chiplets, the data locality should be taken into consideration to avoid unnecessary inter-chip communication. The data transfer on the package behaves differently from that on the chip. On-package transfer exploits a 2D-mesh NoP [64] or a shared IO die [58] with higher complexity and overhead than the on-chip communication. Naturally, the inter-chip communication raises another problem: how to decide the chiplet granularity in a multichip system? With the required number of MAC units, distributing them to different numbers of chiplets brings a trade-off between communication overhead and chiplet area. These problems motivate us to develop a systematic tool that helps architects explore superior designs and helps developers map DNN workloads efficiently.

III. Baseline Architecture: Multichip Hierarchical Model

To analyze the workload mapping, we first introduce a universal hardware model shown in Figure 2. Later, we analyze the inefficiencies of the dataflow in Simba and provide a hierarchical output-centric dataflow that is typically more friendly to multichip accelerators.

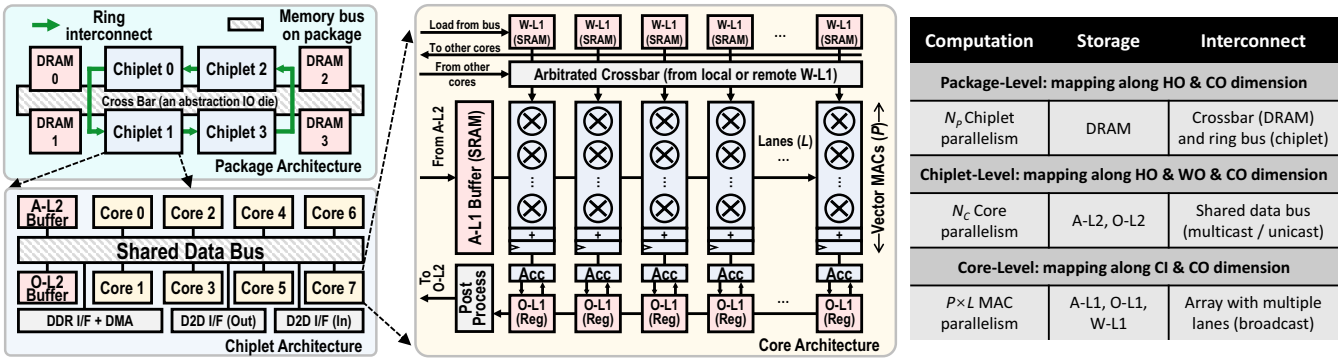
TABLE I: The energy overhead and characters of typical operations in a 16 nm multichip hardware system. The die-to-die energy is from GRS [66].

Operation	Energy (pJ/bit)	Relative cost	Feature
DRAM access	8.75	364.58×	As a slaver attached to a standard high-speed bus and transfer data to the chiplet through DDR PHY
Die-to-die communication	1.17	53.75×	Go through a pair of D2D PHYs to transfer data from chip to chip
L2 access (32KB SRAM)	0.81	33.75×	SRAM multicast or unicast data, which helps to save time and energy
L1 access (1KB SRAM)	0.3	12.5×	
Register read-modify-write	0.104	4.3×	Frequently accessed in WS dataflow
8bit MAC	0.024	1×	Energy is decided by utilization

A. Hierarchical Parallel Model

1) Core Architecture: We define an accelerator core as a PE array connected with several local buffers for activations (A-L1), weights (W-L1) and outputs (O-L1). The core architecture is a Simba-like design that contains L lanes of P-size parallel vector MAC units with weight-stationary dataflow. The output-channel and input-channel are mapped along the L and P dimensions of the PE array. The A-L1 and W-L1 buffers are generated with double SRAMs to overlap the data loading and computation time. As for the O-L1 buffer, we implement it with registers to complete the read-modify-write operation in one cycle. Besides, during once workload allocation, an $\mathrm{HO_C}\times\mathrm{WO_C}\times\mathrm{L}$ output tile is assigned to the core. With the given $\mathrm{HO_C}\times\mathrm{WO_C}\times\mathrm{L}$, we can figure out the local memory footprint requirements of various buffers.

2) Chiplet Architecture: We model the chiplet-level architecture with $\mathrm{N_C}$ cores, a shared activation buffer (A-L2), and a global output buffer (O-L2). All cores and buffers are interconnected via a central bus attached to interfaces for the off-chip memory and remote chiplet communication. We employ the ground-referenced-signaling (GRS) as the die-to-die (D2D) interface model that consumes 1.17 pJ/bit in 16 nm process technology [66]. The central bus can multicast data from the A-L2 buffer to the lower-level cores, conducive to the activation-shared workload partition. The O-L2 is responsible for collecting the final results of all cores in the once workload assignment and then writes back to DRAM. The function of A-L2 is to reuse activations of halo regions when adjacent planar tiles are assigned to several cores. For weights, we organize all W-L1 into a buffer pool and reallocate them in different sharing modes. For example, when core-1 and core-2 require the same weights, their W-L1 buffers are merged into a shared group and broadcast the data to two PE arrays. Otherwise, two cores are distributed with the private W-L1 buffers, and each PE array can only access the local W-L1 space. This policy



Computation	Storage	Interconnect
Package-Level: mapping along HO & CO dimension		
N_P Chiplet parallelism	DRAM	Crossbar (DRAM) and ring bus (chiplet)
Chiplet-Level: mapping along HO & WO & CO dimension		
N_C Core parallelism	A-L2, O-L2	Shared data bus (multicast / unicast)
Core-Level: mapping along CI & CO dimension		
$P \times L$ MAC parallelism	A-L1, O-L1, W-L1	Array with multiple lanes (broadcast)

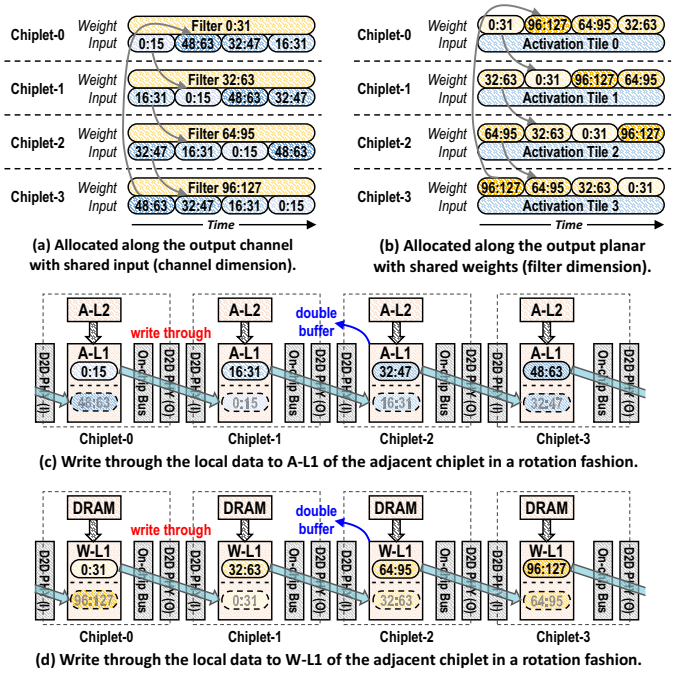
Fig. 2: Overview of the three-level architecture for the multichip accelerator.

refrains from duplicating weights and makes full use of the on-chip W-L1 buffers.

3) Package Architecture: We integrate N_P chiplets via a simple directional ring network on the package to simplify the inter-chip communication. Chiplets on the package are attached to N_P global DRAMs via a crossbar that allows chiplets to access the whole off-chip memory space. Based on the directional ring network, we introduce a rotating transfer shown in Figure 3 for sharing data among several chiplets. Generally, each input feature map is shared by all filters for various output channels. Therefore, when chiplets are mapped along the output channel dimension, they require the same input activations and distinct weights. In this case, each chiplet is allocated with $1/N_P$ input channels to process the corresponding computation (Figure 3(a)). Subsequently, each chiplet writes through its buffered activations from the local A-L1 buffer to the adjacent chiplet via the ring interconnection (Figure 3(c)). Such operations repeat for N_P times to complete the accumulation of all input channels. Likewise, when chiplets process diverse output feature map tiles, they transfer weights and keep activations stationary on the chip (Figure 3(b), (d)). This rotating transfer strategy provides a straightforward solution for chiplets to spatially share data and avoids duplication.

B. Dataflow

As shown in Figure 4(c)∼(d), Simba adopts a WS-based weight-centric dataflow and performs the convolution computation in a systolic-like fashion on the NoC and NoP [55, 78]. The weight-centric dataflow means that the spatial mapping (parallel_for) centers around the weight dimension (e.g., CI and CO). Simba splits input channels (CI) along rows and splits output channels (CO) along columns. However, the weight-centric dataflow fails to leverage the planar dimension (HO and WO) of the activations, leaving the hidden overhead of reloading the halo regions. Besides, partial sums are accumulated from the top to the bottom row across cores and chips, which leads to higher communication overhead derived from partial sums with a wider bit-width (24 bits compared to 8 bits for inputs and weights). Additionally, to transfer 8-bit inputs and weights and 24-bit partial sums using a unified interface, Simba employs a high-cost SerDes IP in the PE router.



(a) Allocated along the output channel with shared input (channel dimension).

(b) Allocated along the output planar with shared weights (filter dimension).

(c) Write through the local data to A-L1 of the adjacent chiplet in a rotation fashion.

(d) Write through the local data to W-L1 of the adjacent chiplet in a rotation fashion.

Fig. 3: The rotating transfer for data (activations and weights) sharing via the directional ring network on the package. (a)∼(b) present the dataflow for activation and weight sharing. (c)∼(d) show the hardware mechanism of the rotating transfer using the write-through manner.

In consequence, as shown in Figure 4(a)∼(b), we apply an OS-based output-centric dataflow to the package and chiplet level by mapping the workload along the output dimension (CO, HO, and WO). Each core does not export results until all the partial sums have been accumulated and then re-quantized to 8-bit data for the next layer. In this way, only the 8-bit inputs and weights are transferred through the simplified interfaces with lower communication overhead. Based on the output-centric dataflow, We further introduce (1) the spatial primitive to represent the workload partition on the package and chiplet (similar to parallel_for in Simba), (2) the temporal primitive to describe the loop unrolling strategy (similar to for in Simba), and (3) the rotating primitive to describe the data sharing among chiplets. The spatial-temporal pair generates

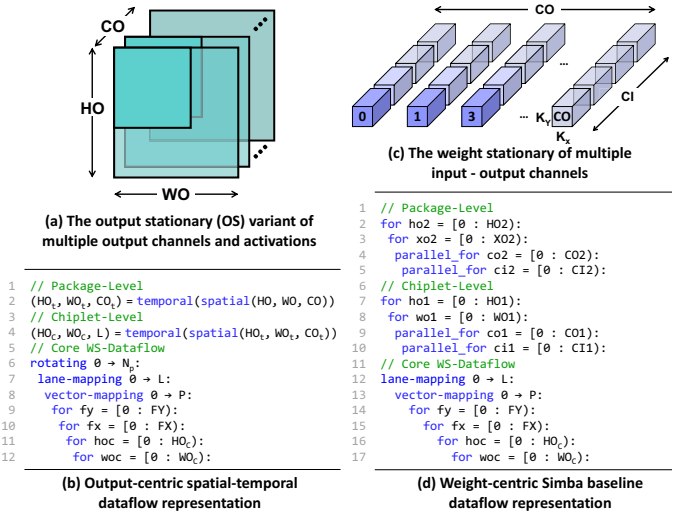


Fig. 4: (a)∼(b) show the employed OS dataflow and the proposed output-centric spatial-temporal dataflow representation. (c)∼(d) present the Simba baseline weight-centric dataflow. Some necessary notations can refer to Figure 5(f).

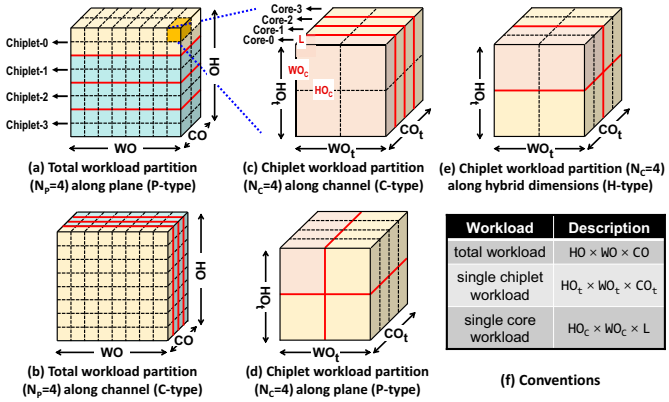


Fig. 5: The visualization of loop transformation using spatial and temporal primitives on the package and chiplet. Cubes in the figure represent output cubes. The bold red lines denote the spatial partition for several chiplets or cores, and dotted lines denote the temporal partition for a single workload for a chiplet or core each time.

a single workload for chiplets or cores each time. It provides excellent convenience for describing the workload mapping and evaluating the memory access overhead, which will be introduced in Section IV. Note that the rotating transfer occurs in the A-L1 or W-L1 buffers (Figure 3(c)∼(d)), so we place the rotating primitive inside the core-level block to complete the computation of an output tile.

IV. Hierarchical and Analytical Framework for Multichip DNN Accelerators

This section first introduces a framework with the output-centric dataflow description and the C^{3}P evaluation methodology. In addition, we analyze the preferred choice of the partition pattern when we apply the spatial primitive. Based on the framework, we then present a detailed characterization of the proposed NN-Baton.

A. The Hierarchical Output-Centric Dataflow Description

Loop transformation is the core of dataflow analysis, which has been frequently studied in these years [32, 41, 49, 71]. However, they need to analyze intricate tiling and unrolling strategy for the whole seven-dimensional loop nest. In the following, we intend to use the spatial and temporal primitives to describe the loop tiling and loop unrolling, respectively.

1) Spatial Primitive: It refers to the spatial parallelism mapping on the computation units (similar to parallel_for in Simba). Figure 5 shows the spatial partition strategies, two for the package level, and three for the chiplet level. In the package level, an output cube is spatially divided into N_P parts (Figure 5(a)∼(b)) along the channel or plane dimension. The P-type partition is more likely to be applied in the large feature map layer to parallelize the plane dimension with shared weights. Given that the large feature map layers usually show up early in a model with a small number of weights, chiplets share weights cost-efficiently via ring network using the rotating transfer. Similarly, The C-type partition is adaptive to the wide layers with intensive weights and fewer activations that can be shared among chiplets at a low cost. Additionally, for the limited on-chip memory capacity, each time a chiplet is delivered with a single chiplet workload of $HO_t \times WO_t \times CO_t$.

Owing to the lower on-chip communication cost, we introduce a supplemental partition, H-type that allows to partition along both dimension simultaneously (Figure 5(c)∼(e)). The chiplet-level spatial primitive provides higher flexibility than that on the package. At this point, two-level spatial primitives provide six loop tiling combinations in total. With a particular partition combination in a layer, the organization of W-L1 buffers, the central bus mode for data sharing, and the transfer path for die-to-die sharing are then reconfigured.

2) Temporal Primitives: It refers to the temporal unrolling mapping in sequence (similar to for in Simba). Temporal primitives target the loop unrolling following the spatial partition of macro workloads. The optimal strategy usually depends on the layer characteristics. Thanks to the output-centric dataflow, we reduce the unrolling search space from the seven-dimensional loop nest to only two dimensions: the channel-priority (C dimension in the inner-loop) unrolling and the plane-priority (H-W dimension in the inner-loop) unrolling. Figure 6(a) illustrates the alternative choices with two temporal primitives in the hierarchy for generating four possibilities in total. The channel-priority unrolling is friendly to the weights reuse if the W-L1 buffers can hold the total weights for the single chiplet/core workload computation. On the contrary, activation reuse potentially gains benefit from the plane-priority unrolling. Combined with six spatial partition manners, our dataflow space provides twenty-four loop transformation solutions in our hierarchical framework that facilitates the analysis in Section IV-B.

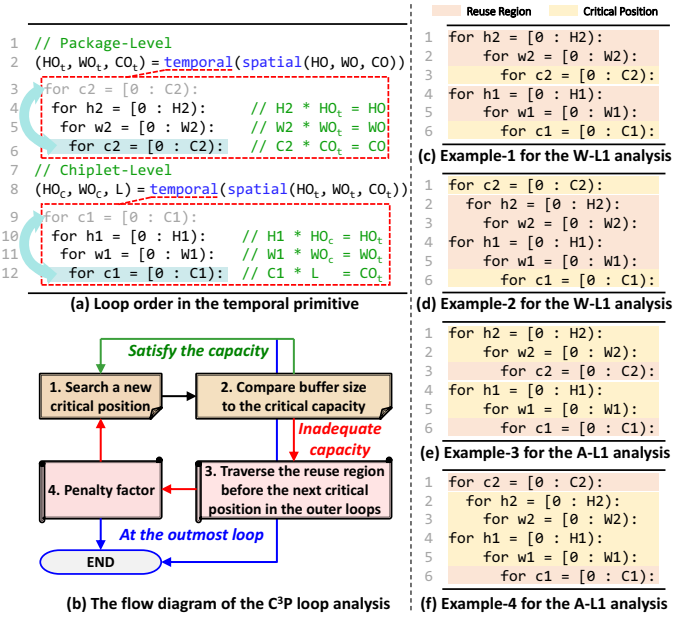


Fig. 6: (a) presents loop unrolling (plane-priority or channel-priority) inside the temporal primitives. (b) provides the C^{3}P flow diagram to calculate the total memory access for a specific temporal combination. (c)∼(f) show examples for the memory access analysis of W-L1 and A-L1 buffers.

B. The Analytical Methodology for Evaluation

The DNN workload mapping strategy and the buffer sizes of each level in the memory hierarchy are two principal factors for memory access. To concisely evaluate the memory access overhead, we formulate a Critical-Capacity Critical-Position (C^{3}P) methodology to understand the relation between the buffer size and workload mapping. The critical positions (Cp) denote the loops that decide the data reuse for a buffer, while the critical capacity (Cc) refers to the buffer sizes that allow reusing data for the outer-loop.

Figure 6(c)∼(d) show examples of W-L1 analysis. We divide temporal loops into two categories: the relevant and irrelevant loops for weights. The relevant loops decide the critical positions for W-L1 while the irrelevant loops between critical positions (or the boundary) form several reuse regions. After that, we leverage the flow diagram shown in Figure 6(b) to analyze the example-1. Firstly, the loop C1 is the first critical position Cp_1, and the corresponding Cc_1 is the capacity to reuse data for the outer-loop, W1-H1. Then, we can obtain the value of the first critical point: $\mathrm{Cc}_1 = \mathrm{C1} \times \mathrm{filters}$, where filters refers to the volume of weights used for a single core workload. W-L1 with less than Cc_1 size will encounter $\mathrm{H1}\times\mathrm{W1} - 1$ access penalties reloading from the off-chip memory. Later, we search the next Cp in the outer-loops and perform the same procedures. Just note that the second critical capacity can be calculated as $\mathrm{Cc}_2 = \mathrm{C2} \times \mathrm{C1} \times \mathrm{filters}$. The example-2 in Figure 6(d) is similar, but the minimal capacity without penalty only depends on Cp_1 because Cp_2 is at the boundary of the loop nest. Given that several W-L1 buffers can be merged and shared for multiple cores, the actual buffer size of W-L1 may vary with layers in different sharing modes.

Figure 6(e)∼(f) present the example for the A-L1 buffer. The example-3 shows another loop case where the first loop is a reuse region. The lower degree of these six loops is the PE array computation for the $\mathrm{HO_C}\times\mathrm{WO_C}\times\mathrm{L}$ workload. Therefore, we additionally supplement Cp_0 and Cc_0 to complete the framework. Besides, different from the analysis of W-L1, the critical capacity for a specific input feature map is determined by the workload size, kernel size, and stride. The example-4 in Figure 6(f) is a bad case for A-L1 because Cc_1 does not contribute to any data reuse. Only when the buffer size is larger than Cc_2 does the buffer locality come into effect. The A-L2 analysis is quite similar to A-L1 with the additional operation to sum up the entire on-chip workload and then calculate the corresponding input feature size.

In summary, illustrated in Equation(1)∼(2), the total memory access of the j-level memory ($A_{tot}^{(j)}$) comprises two parts: the intrinsic access (A_0) decided by the layer configuration and the total penalty access. $\mathbf{N}_j$ denotes the set of critical positions ranging from the outmost critical position to the j^{th} one (i.e. Cc_j, Cc_{j+1}, ...). It is observed that buffers with large enough capacities contribute to no access penalty at high energy and area overhead costs. The $A_{tot}^{(j)}$ can be calculated as:

$$A_{tot}^{(j)} = A_0 \times (1 + \prod_{k \in \mathbf{N}_j} P_k) \tag{1}$$

where P_k is defined as Equation (2). $\mathbf{R}_k$ denotes the reuse region between the k^{th} and $(k+1)^{th}$ critical position. Besides, LC means the loop count. With the aforementioned C^{3}P methodology, we can evaluate the overhead of each memory level in the chiplet architecture.

$$P_k = \begin{cases} \prod_{i \in \mathbf{R}_k} LC_i, & \mathrm{buf} < \mathrm{Cc}_k \\ 1, & \mathrm{buf} \geqslant \mathrm{Cc}_k \end{cases} \tag{2}$$

C. Feature Map Partition Pattern Analysis

During the previous analysis, we combined the H and W dimensions in the dataflow description and ignored the

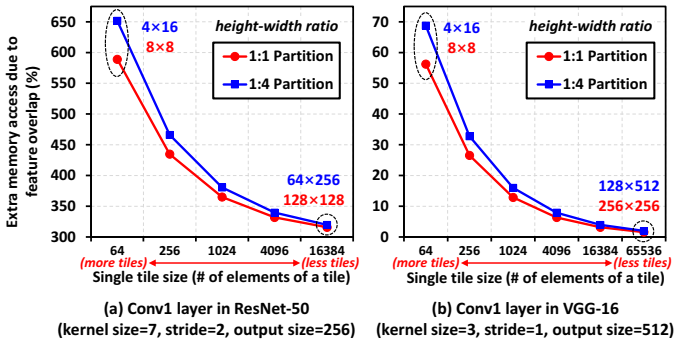


Fig. 7: The redundant memory access with 1:4 and 1:1 partitions patterns in two convolution layers derived from the input feature overlap when workloads are allocated to several chiplets or cores. The input resolution of model is 512×512.

partition pattern. Previous studies for the single-chip accelerator adopted various partition patterns, including stripe [28], rectangle [7], and square [36]. However, with the same number of elements, the spatial partition pattern for H and W dimensions can significantly impact memory access, especially in a multichip system. In a multichip system for large-scale computation (e.g., 512×512 inputs), an inappropriate pattern can lead to unexpected energy overhead originated from halo regions. Consequently, the partition pattern selection should be taken into consideration in the workload mapping procedure.

Figure 7 shows the redundant memory access results from the planar partition with different patterns in two convolution layers. The first convolution layer in ResNet-50 features 7×7 kernel size and 2×2 stride, leading to halo regions of five elements on each side and up to 650% memory access increase. It is observed that the square pattern enjoys less redundant access compared to the rectangle (stripe) one, but the gap between them tends to be smaller when the tile size is getting larger. Compared to the 7×7 convolution, the 3×3 convolution in VGG-16 presents lower extra access while they have the same variation trend.

The conclusions in Figure 7 motivate us to employ a square pattern as far as possible to reduce redundant memory access, especially in the temporal partition. As shown in Figure 5, the temporal primitive generates numerous small tiles, and thus the square pattern is a preferred choice in this scenario. We observe that the package-level spatial primitive generates only N_P tiles (1∼8 in this paper), and theoretically, both choices seem acceptable. However, in the multichip system, to provide enough bandwidth for four chiplets, four DRAMs are integrated into the system. An appropriate data layout is indispensable to avoid memory access conflict and ensure processing efficiency. Figure 8(a) shows a massive central halo region when we use the square pattern, indicating that the central data need to be accessed by four chiplets. In contrast, the halo data are accessed by two chiplets at most in the rectangle pattern (Figure 8(b)). As for the on-chip spatial partition, the control logic can be more flexible so that the preferred pattern generally depends on the specific design and the shape of a single chiplet workload.

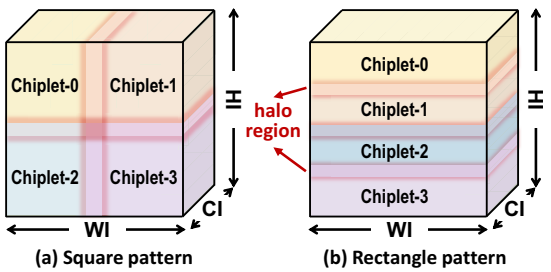


Fig. 8: The visualization of the halo regions using two partition patterns (square and rectangle). The overlap regions potentially lead to the DRAM access conflict.

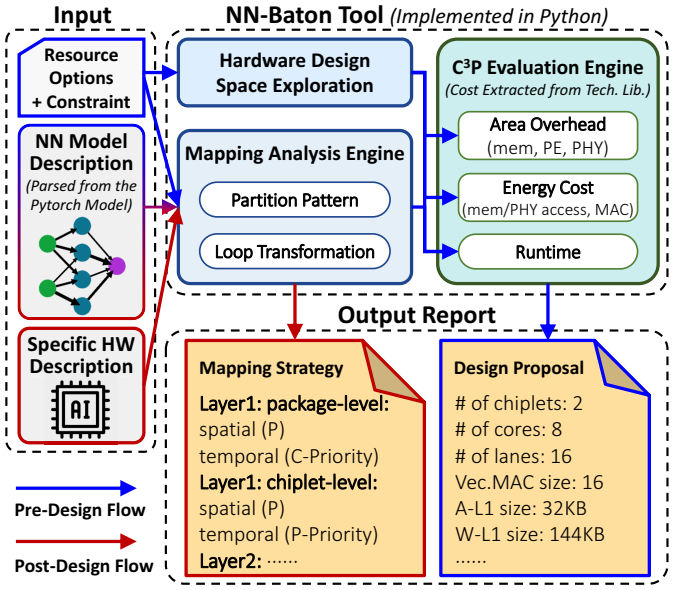


Fig. 9: An overview of the NN-Baton diagram composed of three key components: hardware design space exploration, mapping analysis, and cost analysis modules.

D. The Proposed NN-Baton Automatic Tool

Figure 9 provides a high-level description of the proposed NN-Baton automatic tool. NN-Baton can generate the workload mapping strategy and the chiplet granularity in a pre-design flow and a post-design flow, respectively.

The pre-design flow helps architects decide the chiplet granularity and choose an appropriate hardware resource scheme for computation and memory with the given neural network workloads and hardware constraints. The constraints primarily include the number of MAC units and the area budgets. Provided with mapping strategies and configured with a target process technology, the C^{3}P evaluation engine estimates the area and energy of different possible hardware design choices. The runtime estimation is decided by the total number of MAC units and the utilization. The hardware with too high channel-wise parallelism (i.e., L, the number of lanes) is improper for the thin layer, leading to the under-utilization of computing resources. Finally, the optimal proposal is reported in the output, followed by the C^{3}P evaluation engine.

The post-design can be viewed as a sub-process of the pre-design flow with a specific hardware configuration. This flow produces a detailed mapping strategy for deploying the model on hardware with spatial and temporal primitives. The spatial primitives contain the partition dimension and the partition pattern, while temporal primitives contain the loop order and loop counts. The reported information can be potentially used for the optimization of the hardware compiler.

V. Experimental Setup

A. Hardware Configurations

We model a multichip DNN accelerator based on the description of Section III with 8-bit arithmetic and 24-bit reserved width for partial sums. To evaluate the area and energy

overhead, we use standard cells and memory models from ARM and synthesize a computation core in the UMC 28 nm process technology using Synopsys Design Compiler [60]. Scaled to 16 nm technology to match the GRS macro, the area and power of an 8-bit MAC are respectively 135.1 um^2 and 0.024 pJ/op running at 500MHz frequency.

We select the appropriate multiplexer width and number of banks with a given SRAM configuration for the optimal area and power from the memory model library. As demonstrated by several preliminary evaluations, the area and power approximately satisfy a linear relationship with the SRAM size presented in Figure 10, which allows us to extend the exploration space of memory search using linear regression. The average GRS energy in our evaluation is 1.17 pJ/bit with a 0.38 mm^2 area [66]. The DRAM power is estimated to be 8.75 pJ/bit [22, 71]. The total area of a chiplet includes SRAM, RF, MAC units, and the off-chip PHY and ignores the controller and other IP modules.

B. Workloads

From state-of-the-art DNN models [20, 31, 53, 57], we can extract some representative layers as follows: activation-intensive layers (activations<weights), weight-intensive layers (activations>weights), large kernel-size layer (7×7), point-wise layer (1×1), and other common layers (3×3). We also select AlexNet, VGG-16, ResNet-50 and DarkNet-19 with the input resolutions of 224×224 (for classification tasks) and 512×512 (for detection tasks). AlexNet contains convolution layer of diverse kernel sizes, ranging from 3×3 to 11×11 while the other three models mainly contain 3×3 and 1×1 layers. ResNet-50 and DarkNet-19 are wide models with up to 2048 channels. The feature map size in ResNet-50 reduces earlier than that in VGG-16 and DarkNet-19. Consequently, compared to ResNet-50, their peak memory requirements (show up in Layer-1∼2) for activations are four times as many.

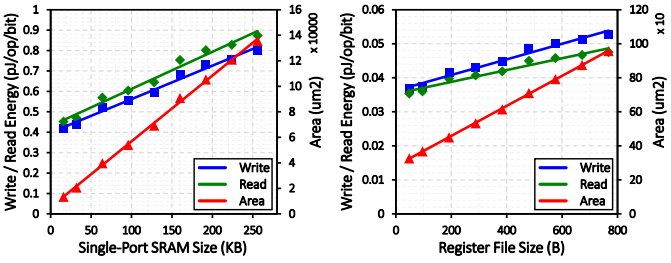


Fig. 10: The linear relationship between the memory (SRAM and Register File) size and overhead (scaled to 16 nm). The energy of RF refers to one read-modify-write operation.

TABLE II: The design space of computation resources and memory footprint for the experimental setup.

Design Space Exploration			
Computation Resources		**Memory Footprint**	
Vector-MAC (P)	2, 4, 8, 16	**O-L1 Size (B)**	48 - 144
# of Lanes (L)	2, 4, 8, 16	**A-L1 Size (KB)**	1 - 128
# of Cores (N_C)	1, 2, 4, 8, 16	**W-L1 Size (KB)**	2 - 256
# of Chiplets (N_P)	1, 2, 4, 8	**A-L2 Size (KB)**	32 - 256

C. NN-Baton Implementation

We implement NN-Baton based on Figure 9 in Python. The resource options from Table II establish the whole exploration space for our evaluations. The O-L2 buffer size is set to match the volume of the final elements of a single chiplet workload. The model description is parsed from the Pytorch model using the `torch.jit` function [51]. The mapping analysis engine adopts exhaustive search to evaluate hundreds of cases, including partition patterns with different height-width ratios and loop transformation of various spatial-temporal combinations introduced in Section IV. Based on the extracted area per unit and energy per operation in Section V-A, the C^{3}P evaluation engine models the corresponding area and energy of the accelerator. Runtime depends on the total MAC units and the computation resource utilization. We establish a simulator to obtain the runtime for a specific workload. In the next section, we employ the pre-design and post-design flow to discuss the workload mapping and design space exploration for the multichip accelerator.

VI. Case Studies

A. Workload Mapping on the Multichip Accelerator

1) Analysis of Mapping Diverse Layers: We first configure the hardware model with 4 chiplets, 8 cores, 8 lanes of 8-size vector MAC, 1.5KB O-L1, 800B A-L1, 18KB W-L1 and 64KB A-L2. With the given hardware, we employ NN-Baton to explore the workload mapping for five distinct layers using different spatial partition strategies. We use the VGG-16 conv1, VGG-16 conv12, ResNet-50 conv1, ResNet-50 res2a_branch2a and res2a_branch2b as the activation-intensive layer, weight-intensive layer, large kernel-size layer, point-wise layer and common layer, respectively. From the results in Figure 11, we observe that the hybrid partition in the chiplet-level ((C, H) or (P, H)) provides the overall lower energy overhead. In the activation-intensive and large kernel layer, the feature map size and the halo region tend to be large, so P-type partition provides fewer redundant memory access derived from the halo regions. However, in the weight-intensive and especially the point-wise layer, the channel dimension has become the principal bottleneck in the computation, prompting the C-type partition to be a preferable solution. Because we use ResNet-50 res2a_branch2b as the common layer where the burden of activation is a little less than that of weight, the results show the C-type partition is more advantageous than the others. The comparison between two rows in the figure demonstrates the similar trends and validates the stability of NN-Baton for different input sizes.

As for the energy breakdown, we observe that the impact of spatial partition manner for the die-to-die overhead is distinct, which is a primary cause of higher total energy. For example, in the activation-intensive layer, the C-type partition on the package leads to a large number of activations transfer among chiplets while the situation is inverse in the weight-intensive

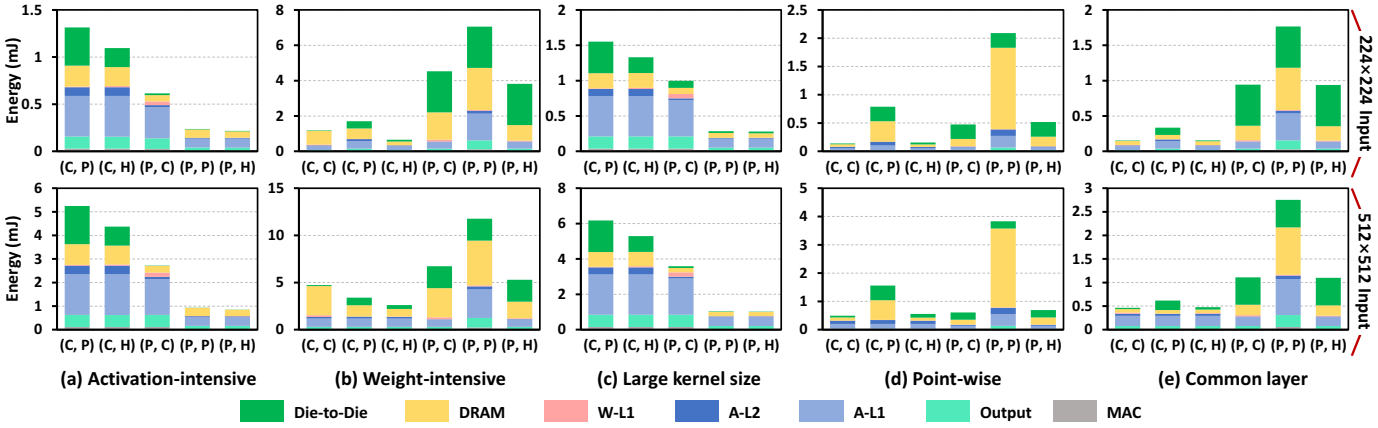


Fig. 11: The energy estimations with breakdown of different spatial partition strategies with two input resolutions are presented in two rows. Five types of layers are listed and we choose the best temporal strategy for each result. In the x-axis, two items denote the spatial partition on the package and chiplet level. C, P, and H denote channel, plane, and hybrid described in Section IV, respectively. We remove the (C, C) option in (a) and (c) due to the mismatch with their small output channels.

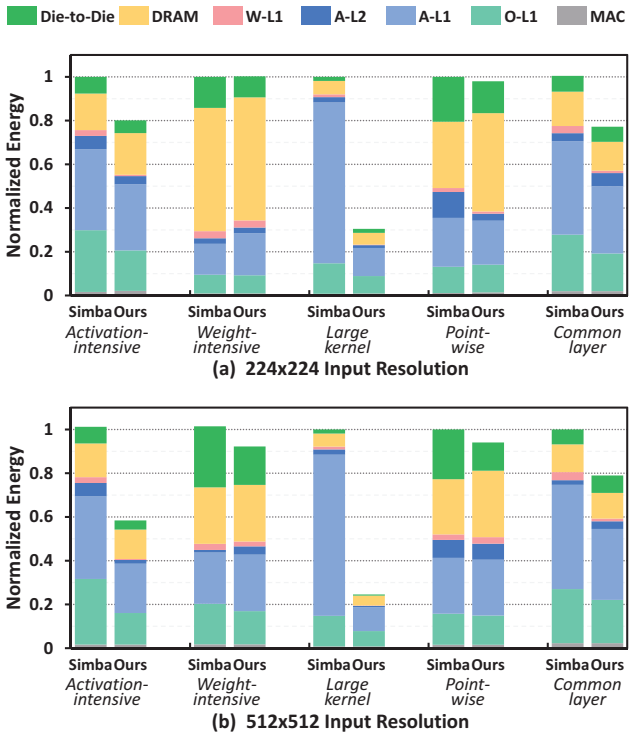


Fig. 12: The normalized energy breakdown of the Simba baseline dataflow and the dataflow generated by NN-Baton in five distinct layers.

layer. The W-L1 energy consumption is always low in the evaluations due to the weight-stationary dataflow in each core. Improper loop transformation for W-L1 primarily leads to extra access to the DRAM.

The diverse preference of different spatial primitives motivates us to apply an optimal solution to different layers properly. Therefore, according to each layer's parameter characteristics, NN-Baton provides a distinct mapping strategy layer-wise to minimize the overall energy cost.

2) Comparison with Simba: We model a 4-chiplet Simba prototype with the configurations (memory sizes and computation resources) and basic dataflow introduced in previous works [55, 78]. For comparability, the multichip accelerator model for NN-Baton is configured with the same memory and computation resources as Simba. Considering that we target the workload mapping optimization in this experiment, like in NN-Baton, we omit the controller and RISC-V overhead of Simba and primarily count the memory write/read operations coupled with the die-to-die communication.

Figure 12 shows the normalized energy in five distinct layers of two input resolutions, and NN-Baton provides overall low energy. We observe significant advantages of NN-Baton in the activation-intensive and large kernel-size layers, especially in the 512×512 resolution case. The advantages are based on the output-centric dataflow and the C-type spatial partition in NN-Baton, beneficial to the activation buffers and DRAM access. These layers contain numerous activations and large size of halo regions, and consequently, the output-centric dataflow can prevent the 24-bit partial sums from transferring on the NoC and NoP and avoid the excessively fragmented H-W dimensions. On the contrary, in layers with smaller feature sizes, such as the weight-intensive and point-wise layers, both perform similarly. Besides, we observe that Simba's die-to-die overhead is always slightly higher than ours due to the massive transfer for partial sums on the package.

Figure 13 provides model-level comparisons with Simba. The results present 22.5%∼44% lower energy cost in three classical models with two input resolutions. According to the previous analysis, Simba baseline dataflow is weak in the layers with large feature maps and halo regions, so the results of 512×512 are always inferior to those of 224×224. The feature map size reduces later in VGG-16 and DarkNet-19 than ResNet-50, so NN-Baton saves more energy in the VGG-16 and DarkNet-19 benchmarks.

These two experiments demonstrate the outperformance of output-centric dataflow compared to the weight-centric

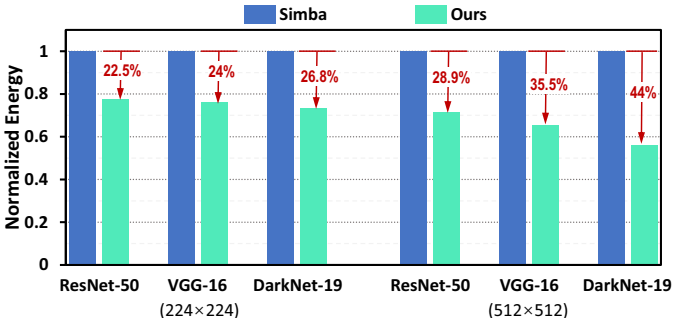


Fig. 13: The comparisons between Simba and NN-Baton using three classical DNN models with 224×224 and 512×512 inputs. The estimation calculates the CONV and FC layers. We reorganize FC layers into point-wise layers for the evaluation.

manner. This is because output-centric dataflow manages to aggregate data more locally to reduce the energy overhead.

B. Design Space Exploration for Multichip Accelerators

1) Analysis of the chiplet granularity: One of the most promising explorations for the multichip accelerator is to decide the granularity of a chiplet. Given a required performance, we need to figure out the optimal number of chiplets that significantly impact the manufacturing costs. With 2048 MAC units in total, there are up to 63 possibilities with different combinations of the number of chiplets per package, cores per chiplet, lanes per core, and MAC units per lane. We assemble the memory hierarchy with buffer sizes proportional to the computation resources in this experiment.

Figure 14 shows the optimal hardware implementation using 1-to-8 chiplets with the best case of the other three computation dimensions when they are deployed to four typical models. We observe that without any area constraint, the energy consumption is generally higher with more chiplets. This is straightforward because on-chip communication generally shows a lower cost than inter-chip communication. With regard to the exceptions in VGG-16 and ResNet-50, the smaller search space of one-chiplet schemes (only three options) limits the exploration for a better optimum. However, with a chiplet area constraint of 2 mm^2, no implementation meets the constraint using one chiplet, and the 4-chiplet implementations provide overall low energy-delay-product (EDP). Encouragingly, even though we select the best case out of 20 implementations in 4-chiplet design, the 4-4-16-8 scheme is always the top-pick one, indicating that this is the optimum computation resource configuration under 2 mm^2 area constraint. The 8-chiplet design shows relatively high energy cost and more runtime because assembling 256 MAC units per chiplets is so scattered that it creates much die-to-die communication overhead.

It is observed that in a multichip accelerator, the chiplet area is a significant driven force to push a design distributed in several dies. Employing the chiplet-based solution sacrifices the performance and energy cost but obtains lower cost and enables the die reuse. Such trade-off is quite common in

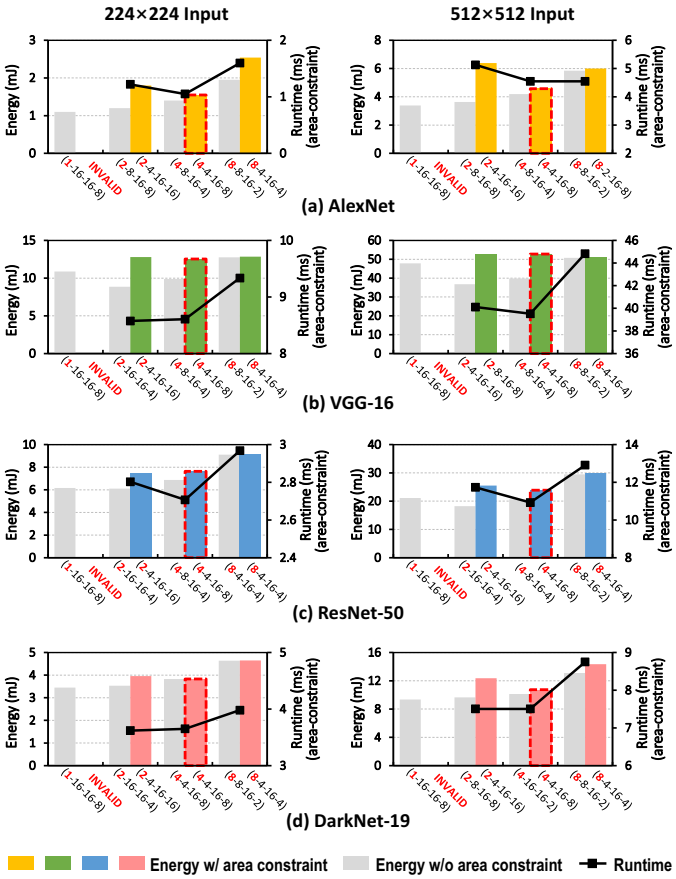


Fig. 14: Different hardware implementations with 2048 MAC units in total. The energy and runtime are estimated with the optimal workload mapping strategy. The grey bars are the best implementations without any area constraint in a specific model. The colored bars are the prime implementations under 2 mm^2 chiplet area constraint. Four groups in a plot refers to the 1, 2, 4, and 8-chiplets implementations. The four-element tuple at the x-axis represents (chiplet, core, lane, vector-size). The bar with a red dotted box enjoys the lowest EDP.

a chiplet-based system, and NN-Baton provides a possible solution for this problem.

2) Design Space Exploration: We employ the NN-Baton pre-design flow to implement the multichip accelerator design space exploration for three models using the parameters listed in Table II. The energy and runtime in each layer are based on the optimal mapping strategy for a specific hardware implementation. Given the area and performance budgets, NN-Baton provides every possible design with the recommended computation and memory allocation. Besides, the search can skip some invalid cases to speed up the space sweeping, such as the A-L1 size smaller than A-L2 or the total MAC units less than the required quantities.

Figure 15 presents the design space exploration with 5800 valid points of over 100,000 sweeping for the whole model. Four colors represent 1, 2, 4, and 8-chiplet implementations, respectively. The grey trend-line divides the results into two zones: the right points are some redundant designs with un-

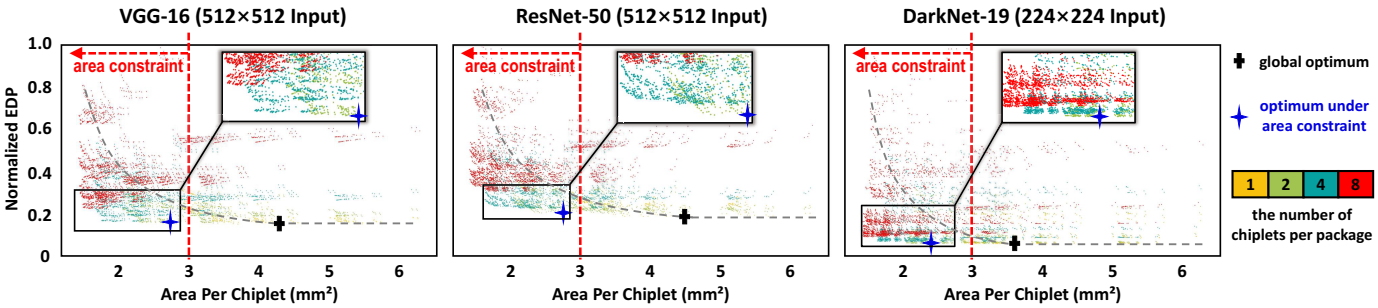


Fig. 15: The design space explorations for the multichip accelerators of 4096 MAC units. With no RISC-V and fewer interface macros in our model than Simba, we apply a chiplet area constraint of 3 mm^2. In the three plots, we zoom in the region of the optimum point under area constraint. Points below the grey trend-lines refer to the more proper designs.

necessary memories, while the left points are potential implementations with appropriate memory allocation. We observe that four colored points appear layered in the plot without area constraint: the 1-chiplet designs (yellow) gather in the lower right region, and the other three gather to the upper left direction in succession. This observation matches the conclusion in Figure 14 that the designs with fewer chiplets sacrifice the area to provide a lower EDP.

Surprisingly, in three benchmarks, the optimal implementations under area constraint are all the configurations of 2-chiplet, 8-core, 16-lane, and 16-vector-size, as the same in Figure 14. We believe that the optimal resource allocation for computing highly depends on the area constraint. However, the memory allocations in these three recommended schemes are distinct. The benchmarks with 512×512 input resolution prefer larger activation buffers (32KB A-L1) than the 224×224 benchmark (4KB A-L1). Also, the W-L1 size in DarkNet of 224×224 input (72KB W-L1) is smaller than the other two benchmarks (144KB W-L1) even though the peak storage of weights in DarkNet (4.5MB) is larger than that of VGG or ResNet (2.25MB). Because the feature sizes in these weight-intensive layers are quite small, the on-chip memory can buffer all the activations without any weight access penalty even if $\mathrm{buf} < \mathrm{C_C}$. In conclusion, the computation resource allocation depends more on the area constraint while memory allocation is sensitive to the target model.

VII. Related Works

A. DNN Workload Mapping

DNN mapping is a necessary procedure when we expect to deploy a specific model on the accelerator efficiently. The loop transformation is required to achieve efficient cache locality for the seven-dimensional loop nest. TeraDeep [17] is a prior work using loop tilling in convolution layers. Later, a series of studies proposed various dataflow for the PE array computation, including output stationary [10, 44], weight stationary [13, 47], input stationary [50], row stationary [6]. For the loop transformation analysis, Zhang *et al.* [76] and Ma *et al.* [42] provided reliable frameworks for the FPGA platform. These loop transformation strategies are all expressed in a loop-nest manner to reveal the scheduling policy [8]. Besides, some variants for domain-specific architectures are proposed recently. TETRIS leveraged a vault-centric partition for NN accelerators with 3D-stacked DRAM [14]. Tangram [15] employed a cascaded loop nest of two layers to describe the layer-pipeline. MAESTRO [32] used a data-centric framework to concisely depict the data movement and reuse behavior in the spatial PE array. These works motivate us that a specific architecture needs the corresponding dataflow description. Currently, there is a lack of a proper description of the chiplet-based system.

B. Hardware Design Space Exploration and Chiplet Design

Hardware design space exploration (DSE) guides the hardware architect to decide the chiplet granularity and distribute resources with the given applications. Caffeine [77] proposed a complete framework with analysis and DSE engines to generate FPGA design using HLS, and AutoDNNchip [68] took a further study on the chip design. Interstellar [71] explored the dataflow and memory hierarchy for the DNN accelerator. MAGNet [61] primarily explored the design space for an NVDLA-like [47] computation core and proposed a novel output stationary - local weight stationary dataflow. Simba [55, 62, 78, 79] is a pioneering work to implement a DNN accelerator using chiplet technology. It provided the basic architecture and dataflow for the multichip accelerator design. Simba also took an in-depth study on the non-uniform workload partitioning on the NoC and NoP. However, they did not provide a complete and analytical framework for the workload mapping and the guidance to the chiplet granularity. When we push DNN accelerators to the chiplet dimension, the granularity of each chiplet is principal for the designers. The existing DSE works mainly focus on traditional platforms like FPGA and monolithic chip. DSE for the multichip accelerators provides a promising future for the architecture community.

VIII. Conclusion

This work filled the gap in the field of DNN workload mapping and DSE for the multichip accelerators. Based on a universal and concise model, we presented a hierarchical and analytical framework with the output-centric dataflow description and the analytical $\mathrm{C^3P}$ methodology for evaluation. We then proposed an automatic tool called NN-Baton to

conduct the orchestration of the DNN workload and guide the chiplet granularity decision. We evaluated NN-Baton relative to Simba and showed a 22.5%∼44% energy reduction in three classical models with 224×224 and 512×512 input resolutions. Besides, we provided a case study about the impact of spatial partition and demonstrated the necessity to employ the preferable spatial primitives in different layers. Finally, we explored the chiplet granularity decision with a required number of MAC units. The results demonstrated that the area constraint is a decisive factor for the optimal computation allocation, while the memory resource allocation typically depends on the NN models.

ACKNOWLEDGEMENTS

This research is supported by the Shannxi Key Research and Development Program of China (Grant No. 2021ZDLGY01-05). We thank the anonymous reviewers for their thoughtful and constructive comments that helped improve the final version of this paper.

REFERENCES

[1] J. Albericio, P. Judd, T. H. Hetherington, T. M. Aamodt, N. D. E. Jerger, and A. Moshovos, "Cnvlutin: Ineffectual-neuron-free deep neural network computing," in *ACM/IEEE Annual International Symposium on Computer Architecture (ISCA)*, 2016, pp. 1–13.

[2] A. Arunkumar, E. Bolotin, B. Y. Cho, U. Milic, E. Ebrahimi, O. Villa, A. Jaleel, C. Wu, and D. W. Nellans, "MCM-GPU: multi-chip-module gpus for continued performance scalability," in *ACM/IEEE Annual International Symposium on Computer Architecture (ISCA)*, 2017, pp. 320–332.

[3] P. Bannon, G. Venkataramanan, D. D. Sarma, and E. Talpes, "Computer and redundancy solution for the full self-driving computer," in *IEEE Hot Chips 31 Symposium (HCS)*, 2019, pp. 1–22.

[4] N. Beck, S. White, M. Paraschou, and S. Naffziger, "'zeppelin': An soc for multichip architectures," in *IEEE International Solid-State Circuits Conference (ISSCC)*, 2018, pp. 40–42.

[5] T. Chen, Z. Du, N. Sun, J. Wang, C. Wu, Y. Chen, and O. Temam, "Diannao: A small-footprint high-throughput accelerator for ubiquitous machine-learning," in *International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2014, p. 269–284.

[6] Y. Chen, J. S. Emer, and V. Sze, "Eyeriss: A spatial architecture for energy-efficient dataflow for convolutional neural networks," in *ACM/IEEE Annual International Symposium on Computer Architecture (ISCA)*, 2016, pp. 367–379.

[7] Y. Chen, T. Krishna, J. S. Emer, and V. Sze, "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," in *IEEE International Solid-State Circuits Conference (ISSCC)*, 2016, pp. 262–263.

[8] Y. Chen, T. Yang, J. S. Emer, and V. Sze, "Eyeriss v2: A flexible accelerator for emerging deep neural networks on mobile devices," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 9, no. 2, pp. 292–308, 2019.

[9] C. Ding, S. Wang, N. Liu, K. Xu, Y. Wang, and Y. Liang, "REQ-YOLO: A resource-aware, efficient quantization framework for object detection on fpgas," in *ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA)*, 2019, pp. 33–42.

[10] Z. Du, R. Fasthuber, T. Chen, P. Ienne, L. Li, T. Luo, X. Feng, Y. Chen, and O. Temam, "Shidiannao: shifting vision processing closer to the sensor," in *ACM/IEEE Annual International Symposium on Computer Architecture (ISCA)*, 2015, pp. 92–104.

[11] H. Esmaeilzadeh, E. R. Blem, R. S. Amant, K. Sankaralingam, and D. Burger, "Dark silicon and the end of multicore scaling," *IEEE Micro*, vol. 32, no. 3, pp. 122–134, 2012.

[12] H. Fang, S. Gupta, F. N. Iandola, R. K. Srivastava, L. Deng, P. Dollár, J. Gao, X. He, M. Mitchell, J. C. Platt, C. L. Zitnick, and G. Zweig, "From captions to visual concepts and back," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 1473–1482.

[13] C. Farabet, B. Martini, B. Corda, P. Akselrod, E. Culurciello, and Y. LeCun, "Neuflow: A runtime reconfigurable dataflow processor for vision," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2011, pp. 109–116.

[14] M. Gao, J. Pu, X. Yang, M. Horowitz, and C. Kozyrakis, "TETRIS: scalable and efficient neural network acceleration with 3d memory," in *Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2017, pp. 751–764.

[15] M. Gao, X. Yang, J. Pu, M. Horowitz, and C. Kozyrakis, "TANGRAM: optimized coarse-grained dataflow for scalable NN accelerators," in *International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2019, pp. 807–820.

[16] R. B. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014, pp. 580–587.

[17] V. Gokhale, J. Jin, A. Dundar, B. Martini, and E. Culurciello, "A 240 g-ops/s mobile coprocessor for deep neural networks," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2014, pp. 696–701.

[18] D. Greenhill, R. Ho, D. M. Lewis, H. Schmit, K. H. Chan, A. Tong, S. Atsatt, D. How, P. McElheny, K. Duwel, J. Schulz, D. Faulkner, G. Iyer, G. Chen, H. K. Phoon, H. W. Lim, W. Koay, and T. Garibay, "A 14nm 1ghz FPGA with 2.5d transceiver integration," in *IEEE International Solid-State Circuits Conference (ISSCC)*, 2017, pp. 54–55.

[19] K. M. Hazelwood, S. Bird, D. M. Brooks, S. Chintala, U. Diril, D. Dzhulgakov, M. Fawzy, B. Jia, Y. Jia, A. Kalro, J. Law, K. Lee, J. Lu, P. Noordhuis, M. Smelyanskiy, L. Xiong, and X. Wang, "Applied machine learning at facebook: A datacenter infrastructure perspective," in *IEEE International Symposium on High Performance Computer Architecture (HPCA)*, 2018, pp. 620–629.

[20] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[21] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A. rahman Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Process. Mag.*, vol. 29, no. 6, pp. 82–97, 2012.

[22] M. Horowitz, "Computing's energy problem (and what we can do about it)," in *IEEE International Conference on Solid-State Circuits Conference (ISSCC)*, 2014, pp. 10–14.

[23] R. Hwang, T. Kim, Y. Kwon, and M. Rhu, "Centaur: A chiplet-based, hybrid sparse-dense accelerator for personalized recommendations," in *ACM/IEEE Annual International Symposium on Computer Architecture, (ISCA)*, 2020, pp. 968–981.

[24] M. James, M. Tom, P. Groeneveld, and V. Kibardin, "ISPD 2020 physical mapping of neural networks on a wafer-scale deep learning accelerator," in *International Symposium on Physical Design (ISPD)*, 2020, pp. 145–149.

[25] 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 *IEEE International Solid-State Circuits Conference (ISSCC)*, 2020, pp. 136–140.

[26] Y. Jiao, L. Han, and X. Long, "Hanguang 800 NPU - the ultimate AI inference solution for data centers," in *IEEE Hot Chips 32 Symposium (HCS)*, 2020, pp. 1–29.

[27] N. P. Jouppi, C. Young, N. Patil, D. A. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden, A. Borchers, R. Boyle, P. Cantin, C. Chao, C. Clark, J. Coriell, M. Daley, M. Dau, J. Dean, B. Gelb, T. V. Ghaemmaghami, R. Gottipati, W. Gulland, R. Hagmann, C. R. Ho, D. Hogberg, J. Hu, R. Hundt, D. Hurt, J. Ibarz, A. Jaffey, A. Jaworski, A. Kaplan, H. Khaitan, D. Killebrew, A. Koch, N. Kumar, S. Lacy, J. Laudon, J. Law, D. Le, C. Leary, Z. Liu, K. Lucke, A. Lundin, G. MacKean, A. Maggiore, M. Mahony, K. Miller, R. Nagarajan, R. Narayanaswami, R. Ni, K. Nix, T. Norrie, M. Omernick, N. Penukonda, A. Phelps, J. Ross, M. Ross, A. Salek, E. Samadiani, C. Severn, G. Sizikov, M. Snelham, J. Souter, D. Steinberg, A. Swing, M. Tan, G. Thorson, B. Tian, H. Toma, E. Tuttle, V. Vasudevan, R. Walter, W. Wang, E. Wilcox, and D. H. Yoon, "In-datacenter performance analysis of a

tensor processing unit," in *ACM/IEEE Annual International Symposium on Computer Architecture (ISCA)*, 2017, pp. 1–12.
[28] S. Kang, J. Lee, C. Kim, and H. Yoo, "B-face: 0.2 MW cnn-based face recognition processor with face alignment for mobile user identification," in *IEEE Symposium on VLSI Circuits*, 2018, pp. 137–138.
[29] Y. Kim, "Convolutional neural networks for sentence classification," in *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1746–1751.
[30] G. G. Ko, Y. Chai, M. Donato, P. N. Whatmough, T. Tambe, R. A. Rutenbar, G. Wei, and D. Brooks, "A scalable bayesian inference accelerator for unsupervised learning," in *IEEE Hot Chips 32 Symposium (HCS)*, 2020.
[31] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems (NIPS)*, 2012, pp. 1106–1114.
[32] 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 *IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2019, pp. 754–768.
[33] M. LaPedus, "10nm versus 7nm," http://semiengineering.com/10nm-versus-7nm, 2017.
[34] M. LaPedus, "Battling fab cycle times," http://semiengineering.com/battling-fab-cycle-times, 2017.
[35] Y. LeCun, "Deep learning hardware: Past, present, and future," in *IEEE International Solid-State Circuits Conference (ISSCC)*, 2019, pp. 12–19.
[36] J. Lee, C. Kim, S. Kang, D. Shin, S. Kim, and H. Yoo, "UNPU: A 50.6tops/w unified deep neural network accelerator with 1b-to-16b fully-variable weight bit-precision," in *IEEE International Solid-State Circuits Conference (ISSCC)*, 2018, pp. 218–220.
[37] H. Liao, J. Tu, J. Xia, and X. Zhou, "Davinci: A scalable architecture for neural network computing," in *IEEE Hot Chips 31 Symposium (HCS)*, 2019, pp. 1–44.
[38] C. Lin, C. Cheng, Y. Tsai, S. Hung, Y. Kuo, P. H. Wang, P. Tsung, J. Hsu, W. Lai, C. Liu, S. Wang, C. Kuo, C. Chang, M. Lee, T. Lin, and C. Chen, "A 3.4-to-13.3tops/w 3.6tops dual-core deep-learning accelerator for versatile AI applications in 7nm 5g smartphone soc," in *IEEE International Solid-State Circuits Conference (ISSCC)*, 2020, pp. 134–136.
[39] M. Lin, T. Huang, C. Tsai, K. Tam, K. C. Hsieh, T. Chen, W. Huang, J. Hu, Y. Chen, S. K. Goel, C. Fu, S. Rusu, C. Li, S. Yang, M. Wong, S. Yang, and F. Lee, "A 7nm 4ghz arm®-core-based cowos® chiplet design for high performance computing," in *IEEE Symposium on VLSI Circuits*, 2019, p. 28.
[40] M. Lin, C. Tsai, K. C. Hsieh, W. Huang, Y. Chen, S. Yang, C. Fu, H. Zhan, J. Chien, S. Li, Y. Chen, C. Kuo, S. Tai, and K. Yamada, "A 16nm 256-bit wide 89.6gbyte/s total bandwidth in-package interconnect with 0.3v swing and 0.062pj/bit power in info package," in *IEEE Hot Chips 28 Symposium (HCS)*, 2016, pp. 1–32.
[41] W. Lu, G. Yan, J. Li, S. Gong, Y. Han, and X. Li, "Flexflow: A flexible dataflow accelerator architecture for convolutional neural networks," in *IEEE International Symposium on High Performance Computer Architecture (HPCA)*, 2017, pp. 553–564.
[42] Y. Ma, Y. Cao, S. B. K. Vrudhula, and J. Seo, "Optimizing loop operation and dataflow in FPGA acceleration of deep convolutional neural networks," in *ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA)*, 2017, pp. 45–54.
[43] C. Meenderinck and B. Juurlink, "(when) will cmps hit the power wall?" in *Euro-Par 2008 Workshops - Parallel Processing*, E. César, M. Alexander, A. Streit, J. L. Träff, C. Cérin, A. Knüpfer, D. Kranzlmüller, and S. Jha, Eds., 2009, pp. 184–193.
[44] B. Moons, R. Uytterhoeven, W. Dehaene, and M. Verhelst, "Envision: A 0.26-to-10tops/w subword-parallel dynamic-voltage-accuracy-frequency-scalable convolutional neural network processor in 28nm FDSOI," in *IEEE International Solid-State Circuits Conference (ISSCC)*, 2017, pp. 246–247.
[45] 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, pp. 1–70.
[46] Nvidia, "Nvidia tesla v100 gpu architecture," https://images.nvidia.com/content/volta-architecture/pdf/volta-architecturewhitepaper.pdf.
[47] Nvidia, "Nvdla deep learning accelerator," http://nvdla.org., 2017.
[48] J. Ouyang, M. Noh, Y. Wang, W. Qi, Y. Ma, C. Gu, S. Kim, K. Hong, W. Bae, Z. Zhao, J. Wang, P. Wu, X. Gong, J. Shi, H. Zhu, and X. Du, "Baidu kunlun an AI processor for diversified workloads," in *IEEE Hot Chips 32 Symposium (HCS)*, 2020, pp. 1–18.
[49] 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, pp. 304–315.
[50] A. Parashar, M. Rhu, A. Mukkara, A. Puglielli, R. Venkatesan, B. Khailany, J. S. Emer, S. W. Keckler, and W. J. Dally, "SCNN: an accelerator for compressed-sparse convolutional neural networks," in *ACM/IEEE Annual International Symposium on Computer Architecture (ISCA)*, 2017, pp. 27–40.
[51] Pytorch, "Torchscript," https://pytorch.org/docs/stable/jit.html.
[52] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 779–788.
[53] M. Sandler, A. G. Howard, M. Zhu, A. Zhmoginov, and L. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 4510–4520.
[54] G. Santoro, G. Turvani, and M. Graziano, "New logic-in-memory paradigms: An architectural and technological perspective," *Micromachines*, vol. 10, no. 6, p. 368, 2019.
[55] 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 *IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2019, p. 14–27.
[56] 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, "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 *IEEE International Solid-State Circuits Conference (ISSCC)*, 2016, pp. 182–183.
[57] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *International Conference on Learning Representations (ICLR)*, 2015.
[58] T. Singh, S. Rangarajan, D. John, R. Schreiber, S. Oliver, R. Seahra, and A. Schaefer, "Zen 2: The AMD 7nm energy-efficient high-performance x86-64 microprocessor core," in *IEEE International Solid-State Circuits Conference (ISSCC)*, 2020, pp. 42–44.
[59] J. Song, Y. Cho, J. Park, J. Jang, S. Lee, J. Song, J. Lee, and I. Kang, "An 11.5tops/w 1024-mac butterfly structure dual-core sparsity-aware neural processing unit in 8nm flagship mobile soc," in *IEEE International Solid-State Circuits Conference (ISSCC)*, 2019, pp. 130–132.
[60] Synopsys, "Design compiler," http://www.synopsys.com/Tools/Implementation/RTLSynthesis/DesignCompiler/Pages.
[61] 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 *Computer-Aided Design, (ICCAD)*, 2019, pp. 1–8.
[62] R. Venkatesan, Y. S. Shao, B. Zimmer, 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 0.11 pj/op, 0.32-128 tops, scalable multi-chip-module-based deep neural network accelerator designed with A high-productivity vlsi methodology," in *IEEE Hot Chips 31 Symposium (HCS)*, 2019, pp. 1–24.
[63] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, "Show and tell: A neural image caption generator," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 3156–3164.
[64] 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 *IEEE International Solid-State Circuits Conference (ISSCC)*, 2020, pp. 46–48.

[65] M. Voogel, Y. Frans, M. Ouellette, J. Coppens, S. Ahmad, J. Dastidar, E. Mohsen, F. Dada, M. Thompson, R. Wittig, T. Bauer, and G. Singh, "Xilinx versal™ premium," in *IEEE Hot Chips 32 Symposium (HCS)*, 2020, pp. 1–46.

[66] J. M. Wilson, W. J. Turner, J. W. Poulton, 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 1.17pj/b 25gb/s/pin ground-referenced single-ended serial link for off- and on-package communication in 16nm CMOS using a process- and temperature-adaptive voltage regulator," in *IEEE International Solid-State Circuits Conference (ISSCC)*, 2018, pp. 276–278.

[67] W. A. Wulf and S. A. McKee, "Hitting the memory wall: implications of the obvious," *SIGARCH Comput. Archit. News*, vol. 23, no. 1, pp. 20–24, 1995.

[68] P. Xu, X. Zhang, C. Hao, Y. Zhao, Y. Zhang, Y. Wang, C. Li, Z. Guan, D. Chen, and Y. Lin, "Autodnnchip: An automated DNN chip predictor and builder for both fpgas and asics," in *ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA)*, 2020, pp. 40–50.

[69] X. Xu, Y. Ding, S. Hu, M. Niemier, J. Cong, Y. Hu, and Y. Shi, "Scaling for edge inference of deep neural networks," *Nature Electronics*, vol. 1, pp. 216–222, 2018.

[70] C. Xue, T. Huang, J. Liu, T. Chang, H. Kao, J. Wang, T. Liu, S. Wei, S. Huang, W. Wei, Y. Chen, T. Hsu, Y. Chen, Y. Lo, T. Wen, C. Lo, R. Liu, C. Hsieh, K. Tang, and M. Chang, "A 22nm 2mb reram compute-in-memory macro with 121-28tops/w for multibit MAC computing for tiny AI edge devices," in *IEEE International Solid-State Circuits Conference (ISSCC)*, 2020, pp. 244–246.

[71] 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 *International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2020, pp. 369–383.

[72] J. Yin, Z. Lin, O. Kayiran, M. Poremba, M. S. B. Altaf, N. D. E. Jerger, and G. H. Loh, "Modular routing design for chiplet-based systems," in *ACM/IEEE Annual International Symposium on Computer Architecture (ISCA)*, 2018, pp. 726–738.

[73] Y. Yu, T. Zhao, K. Wang, and L. He, "Light-opu: An fpga-based overlay processor for lightweight convolutional neural networks," in *ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA)*, 2020, pp. 122–132.

[74] Z. Yuan, J. Yue, H. Yang, Z. Wang, J. Li, Y. Yang, Q. Guo, X. Li, M. Chang, H. Yang, and Y. Liu, "Sticker: A 0.41-62.1 TOPS/W 8bit neural network processor with multi-sparsity compatible convolution arrays and online tuning acceleration for fully connected layers," in *IEEE Symposium on VLSI Circuits*, 2018, pp. 33–34.

[75] J. Yue, Z. Yuan, X. Feng, Y. He, Z. Zhang, X. Si, R. Liu, M. Chang, X. Li, H. Yang, and Y. Liu, "A 65nm computing-in-memory-based CNN processor with 2.9-to-35.8tops/w system energy efficiency using dynamic-sparsity performance-scaling architecture and energy-efficient inter/intra-macro data reuse," in *IEEE International Solid-State Circuits Conference (ISSCC)*, 2020, pp. 234–236.

[76] C. Zhang, P. Li, G. Sun, Y. Guan, B. Xiao, and J. Cong, "Optimizing fpga-based accelerator design for deep convolutional neural networks," in *ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA)*, 2015, pp. 161–170.

[77] C. Zhang, G. Sun, Z. Fang, P. Zhou, P. Pan, and J. Cong, "Caffeine: Toward uniformed representation and acceleration for deep convolutional neural networks," *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.*, vol. 38, no. 11, pp. 2072–2085, 2019.

[78] 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 0.11 pj/op, 0.32-128 tops, scalable multi-chip-module-based deep neural network accelerator with ground-reference signaling in 16nm," in *IEEE Symposium on VLSI Circuits*, 2019, p. 300.

[79] 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.