Computing Utilization Enhancement for Chiplet-based Homogeneous Processing-in-Memory Deep Learning Processors

Bo Jiao*, Haozhe Zhu*, Jinshan Zhang, Shunli Wang, Xiaoyang Kang, Lihua Zhang, Mingyu Wang and Chixiao Chen
State Key Laboratory of ASIC System, Fudan University, Shanghai, PR China
Email:{mywang,cxchen}@fudan.edu.cn

ABSTRACT

This paper presents a design strategy of chiplet-based processing-in-memory systems for deep neural network applications. Monolithic silicon chips are area and power limited, failing to catch the recent rapid growth of deep learning algorithms. The paper first demonstrates a straightforward layer-wise method that partitions the workload of a monolithic accelerator to a multi-chiplet pipeline. A quantitative analysis shows that the straightforward separation degrades the overall utilization of computing resources due to the reduced on-chiplet memory size, thus introducing a higher memory wall. A tile interleaving strategy is proposed to overcome such degradation. This strategy can segment one layer to different chiplets which maximizes the computing utilization. To facilitate the strategy, the modification of the chiplet system hardware is also discussed. To validate the proposed strategy, a nine-chiplet processing-in-memory system is evaluated with a custom-designed object detection network. Each chiplet can achieve a peak performance of 204.8GOPS at a 100-MHz rate. The peak performance of the overall system is 1.711TOPS, where no off-chip memory access is needed. By the tile interleaving strategy, the utilization is improved from 53.9% to 92.8%.

CCS CONCEPTS

• **Computer systems organization** → Neural networks; Data flow architectures.

KEYWORDS

Chiplet, Processing-in-Memory, Memory wall, Domain specific architecture, DNN mapping

ACM Reference Format:
Bo Jiao*, Haozhe Zhu*, Jinshan Zhang, Shunli Wang, Xiaoyang Kang, Lihua Zhang, Mingyu Wang and Chixiao Chen. 2021. Computing Utilization Enhancement for Chiplet-based Homogeneous Processing-in-Memory Deep Learning Processors. In *Proceedings of the Great Lakes Symposium on VLSI 2021 (GLSVLSI '21), June 22–25, 2021, Virtual Event, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3453688.3461499

*The two authors contribute equally. This paper is supported by the Strategic Priority Research Program of Chinese Academy of Sciences under Grant No. XDB44000000, Major Project under Grant No.2021SHZDZX0103, Science and Technology Commission of Shanghai Municipality under Grant No. 19511132000, National Natural Science Foundation of China under Grant No. 62074041, Engineering Research Center of AI Robotics, Ministry of Education and Shanghai Engineering Research Center of AI Robotics.

GLSVLSI '21, June. 22–25, 2021, Virtual Conference

ACM ISBN 978-1-4503-8393-6/21/06…$15.00
https://doi.org/10.1145/3453688.3461499

1 INTRODUCTION

The recent rapid growth of artificial intelligence (AI) algorithms drives a sharp increase of deep learning processors' (DLP) computing power, from few GOPS to tens of TOPS. On the other hand, the computing density of silicon chips is saturating due to the dark silicon phenomenon [1]. Therefore, area expansion becomes the major knob filling the gap. The areas of state-of-the-art DLP chips are approaching the limits of lithography [2], around 700-800 mm^2 under 12-nm FinFET technology. The on-chip SRAM size is also enormous, reaching 192 MB. Unfortunately, it is neither cost-efficient nor robustness-guaranteed to design such huge chips. Given the same probability of defects, every time the chip area doubles, the yield loss doubles as well. In addition, non-recursive engineering (NRE) cost is also escalating and intolerable for mass production of these chips.

There are two technical trends to solve the problem. The first is to increase the energy efficiency of DLPs by using processing-in-memory (PIM) designs [3]. PIM architectures can complete computing inside the weight memories by adopting analog and mixed-signal based computing. The second trend is a chiplet-based integrated system consisting of multiple chips [4]- [6]. The key benefit of the chiplet-based systems is twofold. It is heterogeneous technology-friendly, where processors and mixed-signal IPs can be fabricated under different technology. Also, the system demonstrates good scalability by deploying homogeneous chips of different numbers. Such scalability paves the way to catch up with the AI algorithm speed without the cost and fabrication limits.

Chiplet-based PIM systems, illustrated in Fig. 1, are likely a promising solution of next-generation DLP architectures in terms of both energy efficiency and memory wall. However, efficient mapping strategies for such systems have not been investigated. Compilation strategies for general-purpose many-core chiplet systems are still obscure, although a 4096-core RISC-V chiplet system has already been developed [7]. A cross-layer mapping scheme for chiplet-based digital DLPs is discussed in [8], but is not suitable for PIM scenarios. Moreover, the chiplet hardware requires power-consuming overhead such as high-speed links. On the other hand, state-of-the-art PIM mapping schemes focus on intra- and inter-parallelism to improve neural network utilization [9]. However, their overall performance evaluation is over-optimistic and inadequate for practical PIM hardware implementation.

In this paper, we present a design strategy for chiplet-based PIM systems where a computing utilization enhancement scheme is proposed to overcome the performance loss. The rest of the paper

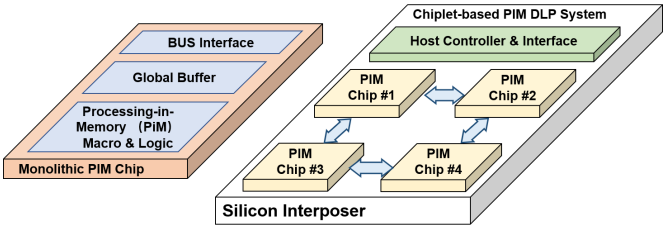


Figure 1: Monolithic vs. Chiplet-based PIM DLP systems

is organized as follows. Section II summarizes the recent related works. Section III introduces the PIM partition techniques and evaluates the performance loss quantitatively. The enhancement scheme, known as tile interleaving, is detailed in Section IV. Section V concludes the paper.

2 BACKGROUND AND RELATED WORKS

Deep learning is a computing and memory-intensive application. Traditional implementations are GPU based, which are energy-inefficient and require large monolithic designs. Recent works explore efficient DLP implementation by investigating chiplet, PIM architectures, and compilation techniques.

Chiplet-based high performance architecture. Unlike monolithic chips, chiplet-based architectures integrate multiple microchip components into one package. Dies are connected via an inter-chip network. Chiplets are used for many-core server CPUs [5] [10], and heterogeneous multi-core system [11]. The chiplet interconnections and the many-core network-on-chip (NoC) are similar. For example, Adapt-NoC in [12] dynamically allocates NOC into different regions.

Simba [6, 8] is a chiplet based scalable DLP architecture. Each chiplet architecture is similar to conventional DLPs, except additional network routers and data links for data transfer. The architecture can support scalable systems from one chip to 36 chips.

PIM-based Non-von-Neumann architecture. The performance of conventional von Neumann architecture based DLPs is limited by the communication bandwidth between the processor and the memory. PIM architectures were first proposed to eliminate this bottleneck by combining computing and storage into one device, such as memristors [13]. The resistance of these cells can be modified by memory write operations and can be performed multiplications with an input voltage according to Ohm's Law.

Another PIM architecture's benefit is energy efficiency. Literature shows that PIM chips have already achieved 10-100x energy efficiency than other DLP implementations [3]. The low-power feature is due to the analog-computing circuits.

Neural network mapping and compilation. Deploying deep neural networks (DNNs) on the domain-specific hardware draws much attention. For systolic array-based accelerators [14], DNNs are commonly converted to matrix-matrix multiplication operation, and the compilers map these operations onto the certain hardware. In addition, edge devices require low power consumption and high energy efficiency, where DDR links might not be available. Dedicated and flexible data flow optimization [15] is adopted to exploit the data reuse of DNNs and relieve the off-chip communication bandwidth.

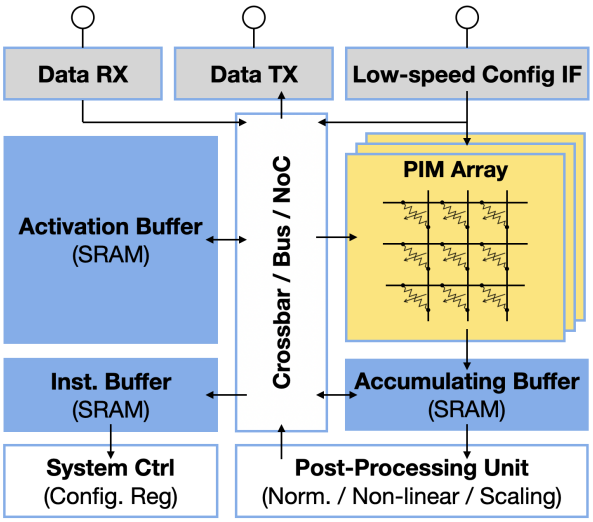


Figure 2: PIM-based DLP architecture.

Compilation strategies are highly hardware-specific. Even though there are open-source general-usage DNN compilation tools such as TVM[16], almost every new DLP architecture requires its own compilation tool. Chiplet-based architectures require communication-aware workload partition [8], while PIM-based architectures support much fewer data flows than conventional DLPs.

3 HOMOGENEOUS PIM PARTITION

This section introduces a baseline chiplet-based PIM system, which is converted from a monolithic PIM processor.

3.1 Monolithic PIM Architectures

Previous DLP research introduced versatile data flow optimization schemes to digital parallel architectures, such as input and row stationary in [17]. These schemes exploit data reuse and greatly compress memory access compared with basic SIMD architectures. Some of these schemes can also be ported to PIM implementations, further pushing the energy efficiency of DLPs [18].

Figure 2 shows a typical monolithic PIM-based DLP architecture, where PIM array calculates matrix multiplication with low power consumption. According to Ohm's Law and Kirchhoff's Laws, the input vector of the PIM array interacts with the pre-stored data via memristor arrays, simultaneously generating multiplication-and-accumulation (MAC) results of multiple columns. Although PIM has high energy efficiency and performance density, its data flow mapping is not as flexible as SIMD architectures. Most non-volatile memory devices used as PIMs, such as ReRAM and flash memory, lack writing endurance, and have long latency for accurate writing. In other words, these cells can only be written within limited times, making it difficult to frequently update these stored data during the neural network inference computing. Therefore, fixed weight placement is widely adopted in the PIM-based DLPs. Data flows used in PIM architectures normally load the weights before computing and keep them read-only during inference. The MAC outputs of the PIM array are added to the intermediate results in an

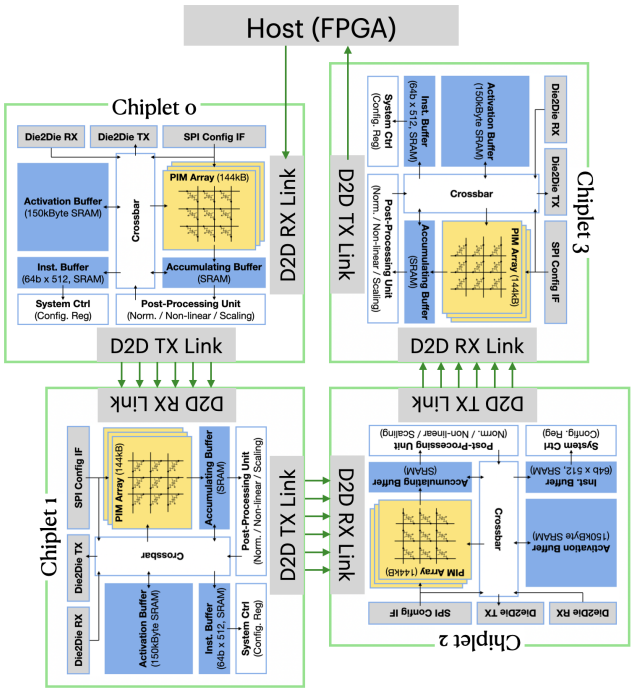


Figure 3: A four-chiplet chiplet-based PIM system.

Accumulating Buffer, and the final MAC results are delivered to the Post Processing Unit for normalization and non-linear activation.

To buffer the input and output activations, the accelerator is also equipped with a large SRAM-based Activation Buffer. Because the weights are fixed, the main memory wall of PIM designs is from these inter-layer results. If there is no off-chip DRAM, these buffer size should be greater than the maximum sum of layer input and output. The typical buffer size is from hundreds of Kilobytes to few Megabytes.

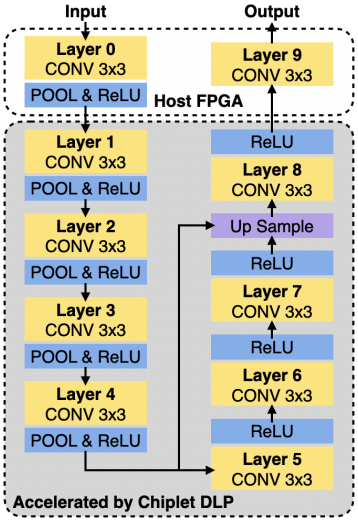
With the explosion of AI algorithms, the monolithic PIM-based DLPs are mainly challenged by massive model weights, especially for the fixed weight placement scheme. More state-of-the-art AI algorithms require hundreds of Mega-byte or even Giga-byte room.

Cost-efficient monolithic DLP chips can hardly afford so many weights of the entire network.

It is unacceptable to continuously expand the area of a single chip, so partitioning large networks into multiple chiplets is expected to meet this challenge.

3.2 Partition from Monolithic to Multi-chiplet

A chiplet-based DLP consists of multiple chips partitioned from the original large DLP. Assuming that all units are identical, we call the system a homogeneous one. Each chiplet has a similar architecture to the monolithic one, but the sizes of PIMs and SRAM buffers are scaled down by the partition. Because all weights are fixed place, the data flow inside the chiplet system is a layer-wise pipeline. In other words, every chiplet is expected to receive data from its predecessor, compute the corresponding layers, and send out the results continuously and simultaneously. A four-chiplet pipeline system is illustrated in Fig. 3.



Layer #	Input Shape $C \times H \times W$	Output Shape $C \times H \times W$	Weight Size (Kb)
1	32x128x64	64x64x32	54
2	64x64x32	64x32x16	108
3	64x32x16	64x16x8	108
4	64x16x8	64x8x4	216
5	64x8x4	128x8x4	432
6	128x8x4	256x8x4	96
7	256x8x4	128x8x4	96
8	256x16x8	128x16x8	864
Total	/	/	1974

Figure 4: A modified Tiny-YOLO for case study.

The PIM chiplet should perform three types of tasks in parallel: a) `RX` task: receive activations from the predecessor chiplets and store them into the local buffer; b) `COMP` task: perform PIM-based convolution or full-connection calculation with the local activations and weights; and c) `TX` task: fetch the output activations from the buffers and deliver them to the succeeding chiplets. Correspondingly, a pair of receiver and transmitter are devised on each chiplet and can work in a duplex.

In a multi-chiplet system, each chiplet can process one or few layers of the entire network. In contrast with the monolithic design, the limited capacity of the SRAM buffers and PIM macros on the chiplet is confronted by more severe memory wall. Furthermore, the chiplet pipeline requires a balanced latency of each chiplet.

Otherwise, the overall data flow might be stalled by unnecessary bubbles. Moreover, the chiplet-based PIM systems need to solve an additional synchronization problem.

The layer-wise pipeline (LWP) can be adopted for chiplet-based DLP systems [9]. It directly divides the network into multiple segments, and the workload of each segment is roughly the same. Each segment is allocated to one chiplet, so the overall system can accelerate the network as a pipeline, which can improve the utilization. In [8], a network of package (NoP) interconnect topology was employed to form a 2D mesh in the chiplet system. This implementation introduces extra router design, increasing both energy and latency.

3.3 Case Study I: A Four-chiplet PIM DLP

In order to quantitatively analyze the performance, we first build a four-chiplet model as Fig. 3, and deploy a real-time object-detection benchmark on it, illustrated in Fig. 4. It is modified from Tiny-Yolo.

Each chiplet includes a 150-kB Activation Buffer and a 144-kB PIM array, and it can perform 1024 MACs per cycle. Thus, it can achieve a peak performance of 204.8GOPS with a clock frequency of 100MHz. Following the LWP mapping strategy, four chiplets are interconnected in a pipeline and equipped with 1.2Gbps/link die-to-die connections. Note that all weights and activations are quantized

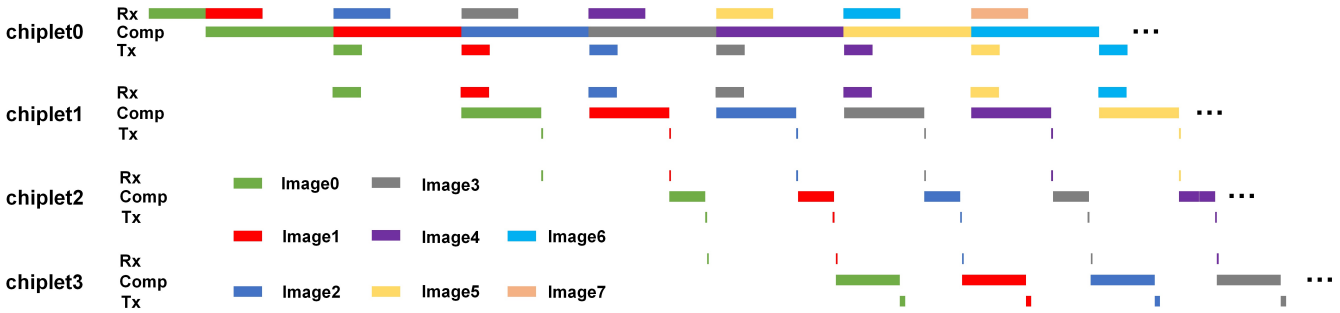


Figure 5: Workload (RX/COMP/TX) breakdown for the four-chiplet PIM system.

Table 1: Four Chiplets Latency Details

Chiplet	Layer	Latency (μs)			Util. (%)
		RX	COMP	TX	
0	1	327.68	1474.56	163.84	100
1	2	163.84	737.28	N/A	62.5
	3	N/A	184.32	40.96	
2	4_1	40.96	92.16	N/A	28.13
	4_2	N/A	92.16	N/A	
	5	N/A	46.08	N/A	
	6	N/A	92.16	N/A	
	7	N/A	92.16	10.24	
3	8	10.24	368.64	54.61	25

to low-bitwidth except the first and last layer. Since PIM circuits only support low-bitwidth MAC, these two layers are deployed on a host FPGA.

The LWP mapping is performed as Table. 1 shows. In practice, the latency of each chiplet is difficult to divide evenly. For most deep neural networks for images, including the adopted one, as the network goes deeper, the number of filters increases so that the weight size of latter layers is much larger than the former ones. Considering that the PIM capacity is limited per chiplet, the chiplets in charge of the latter layers exhaust their PIM capacity more easily than the former ones. For example, chiplet 3 in the case only can be crammed into weights of layer 8. Meanwhile, the trend of the activation size and computing operations is the opposite. As the network goes deeper, the activation size is shrinking sharply due to the pooling function, resulting in a less computing time. Thus, unbalanced latency occurs. Figure 5 shows the practical workload breakdown of the four-chiplet PIM system. The first layer is mapped onto chiplet 0, and the second and third layers are mapped to chiplet 1. The rest layers are mapped on chiplet 2 except layer 8. Thanks to the large weight size of layer 8, the entire chiplet 3 can only map one layer. Detailed analysis shows that the computing utilization of chiplet 2 and 3 is only 28.1% and 25.0%, respectively. It is because the bubbles are inserted due to the unbalanced pipeline latency. The average utilization of the system is only 53.8%. It is necessary to develop a new mapping strategy to enhance the computing utilization for chiplet-based homogeneous PIM DLP systems.

4 TILE INTERLEAVING

As discussed above, the LWP based method should be evolved to meet the chiplet-based PIM systems. In this section, we propose

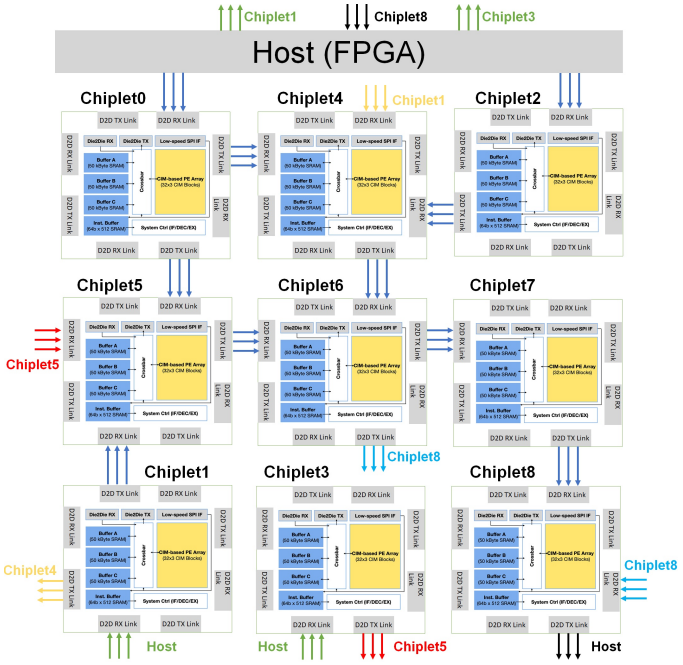


Figure 6: A nine-chip chiplet-based PIM system.

a tile interleaving method without increasing the on-chip PIM or Activation Buffer sizes.

4.1 Tile Interleaving Methods

Observing from Fig. 5, the computing utilization of chiplet 0 and 1 are obviously greater than the rest two. In other words, LWP is a too coarse partition scheme for chiplet-based PIM systems. The basic idea of the interleaving method is to further segment communication data beyond the LWP. Since the utilization of the latter chiplets is saturated by the PIM capacity, the interleaving scheme will be applied to the first few layers.

There are two potential options that can further partition the workload to more chiplets. One is to split the input feature map into smaller tiles, and the other one is to unroll the output channel of the convolution. The latter interleaving method greatly increases the inter-chiplet activation communication, exacerbating the memory

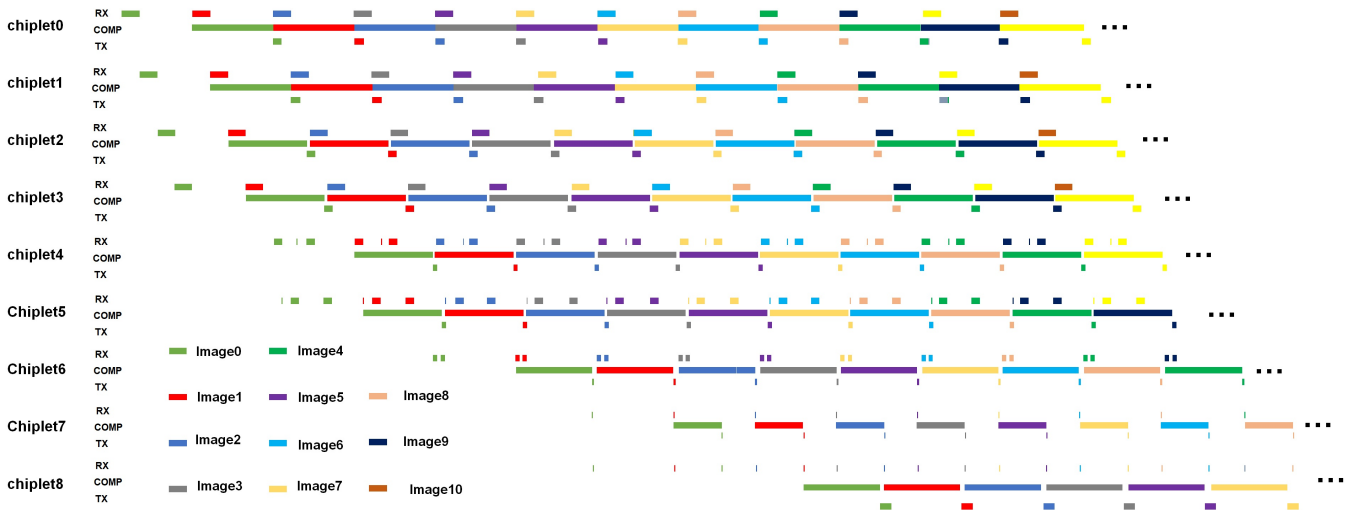


Figure 7: Workload (RX/COMP/TX) breakdown for the nine-chiplet PIM system with tile interleaving.

wall. Therefore, in this paper, we focus on the tile interleaving scheme.

There are four steps to perform the tile interleaving method. First, collect the overall weight capacity, activation capacity, operation latency and data communicating latency of each layer, according to the DNN benchmarks and chiplet hardware specification. In the second place, perform the LWP as a rough and initial segmentation and mapping solution.

The minimum computing latency per chiplet, normally appearing at the last layers, can be evaluated by this step. Thirdly, determine the overall chiplet number and partition/combine those layers of unbalanced latency into multiple/one chiplets by increasing the number of overall chiplets. Layers previously combined on a single chiplet can also be separated onto different chiplets.

Finally, connect the chiplets according to the pipeline flow and validate the overall data flow by simulations. The process can be recursive to find the optimal.

Figure 6 shows a prototype of a chiplet-based PIM system after tile interleaving. The architecture of each chiplet and overall routing fabric is slightly different from the previous implementation. In Fig.3, there are only one-to-one interconnections appearing between two adjacent chiplets, whereas in Fig.6, there are one-to-many and many-to-one interconnections occurring between adjacent and distant chiplets.

At least two extra paths are added per chiplet. A pair of one-to-one TX/RX from the LWP scheme is not sufficient for the tile interleaving chiplet-based PIM system. A straightforward method is to increase the number of TX/RX pairs, as well as the hardware overhead and power consumption. Actually this overhead can be avoided. If time shifts are applied to the chiplet set of the same layer, the different die-to-die paths of would not activated at the same time. In this case, the many-to-one die-to-die interconnection can be implemented as one TX/RX pairs multiplexed in time domain. Note that the scheme should controlled by a top-level synchronization mechanism.

To accommodate the complicated interconnection, a silicon interposer beneath all the chiplet is desired to design in specific for the extra routing.

The compilation tool of the chiplet-based PIM systems should support two functions. First, it should correctly map the partitioned tile data onto the corresponding chiplet, and generate the instructions to perform data transferring and computing. Second, it should control the synchronization among all chiplets. Given that all TX/RX interfaces can only support one-to-one data communication but need to be multiplexed as timing varying, different tiles should be transferred and computed with time shifts. Computing does not starts until the RX data are ready, and neither does the TX. The synchronization mechanism is supposed to be determined as soon as the tile interleaving and partition is completed. To achieve synchronization, a `WAIT` instruction is defined in the system controller instruction set.

4.2 Case Study II: A Nine-chiplet PIM DLP

A nine-chiplet PIM system with tile interleaving in Fig. 6 is evaluated by the same benchmark of Fig. 4. Each chiplet has the same performance specification as mentioned in Section 3.3. In contrast with the four-chiplet PIM system, here is partitioned the first layer onto four chiplets, indexed by chiplet 0 to 3. Similarly, the second layer is segmented onto two chiplets, indexed by chiplet 4 and 5.

Figure 7 also shows the workload breakdown when the same benchmark is compiled and run on the system. Each colored rectangular block represents the corresponding operation of the `RX/TX/COMP`. For chiplet 0-3 and chiplet 7, all of which input activations are from only one source, two adjacent blocks of `RX` in different color mean that two activation maps of two consecutive images are fed into the chiplet one after the other. Among chiplet 0-3, tile interleaving is performed to segment the workload. A constant time shift is inserted to avoid hazards during data transferring. Thanks to the interleaving scheme, some chiplets' data are not from only one predecessor, such as chiplet 4-6 and chiplet 8. For these chiplets, the synchronization should be maintained when different sources are multiplexed. Take chiplet 4's first data receiving as an example, the number of blocks in light green is three, which means its data from three predecessors. It's also worth noting that one chiplet may transmit the output data to multiple chiplets, such as chiplet 0-1 and chiplet 6. It can result from tile-interleaving or the network's

Table 2: Nine Chiplets Latency Details

Chiplet	Layer	Latency (μs)			Util. (%)
		RX	COMP	TX	
0	1	87.04	391.68	43.52	100
1		87.04	391.68	43.52	100
2		84.48	380.16	40.96	97.06
3		84.48	380.16	40.96	97.06
4	2	84.48	380.16	20.48	97.06
5		84.48	380.16	20.48	97.06
6	3	40.96	184.32	N/A	94.12
	4_1	N/A	92.16	5.12	
	4_2	N/A	92.16	5.12	
7	5	5.12	46.08	N/A	58.82
	6	N/A	92.16	N/A	
	7	N/A	92.16	5.12	
8	8	10.24	368.64	54.61	94.12

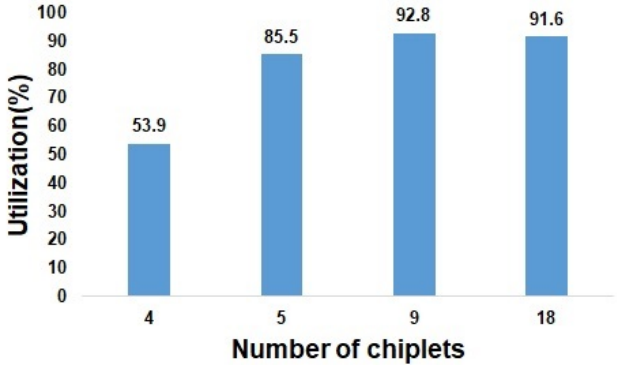


Figure 8: Computing utilization vs. the number of chiplets.

own characteristics. These chiplets transmit data to different destinations continuously. Apparently, the source-chiplet's transmitting and the destination-chiplet's receiving happens at the same time. Also, some chiplets are mapped with more than one layer. For example, chiplet 7 combines three layers and a cross-layer pipeline can be achieved inside the chip.

Data dependency might degrade the pipeline correctness. In the proposed system, the three operations of each chiplet are arranged to cope with data of alternative images at the same time. Therefore, the pipeline bubbles among chiplets can be compressed. Table 2 details the configuration and utilization of each chiplet. Most chiplets' utilization is greater than or close to 95% except chiplet 7. The overall utilization of the system is 92.8%, which improved by 38.9% compared with the LWP only scheme. A complete simulation is performed to evaluate the entire system's performance. Provided that the single chiplet has a peak throughput of 204.8GOPS, the overall system's peak performance can achieve 1.711TOPS.

4.3 Discussion on Chiplet Numbers

As demonstrated above, the chiplet-based PIM system provides a new dimension of DLP design space – the number of chiplets. The computing utilization can be optimized by picking a proper chiplet number, according to the Section 3.3 and 4.2. We also perform the tile interleaving strategy on other chiplet-based PIM systems with different chiplet numbers, As Fig. 8 depicts, it is not true that the more chiplets are used to map the same layer, the higher utilization the system achieves. In fact, when the input feature map is segmented into too many tiles and mapped on too many chiplets, the sum of all chiplets' operation latency is greater than only one chip's operation latency that is mapped with the same layer. In addition, the amount of die-to-die data transferring also increases, causing extra power consumption. For the adopted object detection benchmark, the optimal chiplet number is nine.

5 CONCLUSION

The paper proposes a tile interleaving method for chiplet-based homogeneous PIM DLP systems to enhance the computing utilization. It overcomes the performance loss of previous PIM mapping schemes. This method can segment one layer to multiple chiplets to maximize the overall computing utilization. For a modified Tiny-YOLO benchmark, the utilization of only the LWP mapping method is 53.9%, while the utilization of the tile interleaving method is improved to 92.8%.

REFERENCES

[1] M. B. Taylor, "Landscape of the New Dark Silicon Design Regime," in *IEEE Micro*, vol. 33, no. 5, pp. 8-19, Sept.-Oct. 2013.

[2] Y. Jiao, et al., "a 12nm programmable convolution-efficient neural-processing-unit chip achieving 825tops," in *2020 IEEE International Solid- State Circuits Conference - (ISSCC)*, 2020, pp. 136–140.

[3] Q. Dong et al., "A 351TOPS/W and 372.4GOPS Compute-in-Memory SRAM Macro in 7nm FinFET CMOS for Machine-Learning Applications," IEEE International Solid- State Circuits Conference - (ISSCC 2020), San Francisco, CA, 2020, pp. 242-244.

[4] D. Stow, Y. Xie, T. Siddiqua, and G. H. Loh, "Cost-effective design of scalable high-performance systems using active and passive interposers," in *2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2017, pp. 728–735.

[5] S. Naffziger, K. Lepak, M. Paraschou and M. Subramony, "AMD Chiplet Architecture for High-Performance Server and Desktop Products," in *IEEE International Solid- State Circuits Conference - (ISSCC)*, 2020, pp. 44-45.

[6] B. Zimmer, et al., "A 0.32–128 tops, scalable multi-chip-module-based deep neural network inference accelerator with ground-referenced signaling in 16 nm," *IEEE Journal of Solid-State Circuits*, vol. 55, no. 4, pp. 920–932, 2020.

[7] F. Zaruba, F. Schuiki and L. Benini, "Manticore: A 4096-core RISC-V Chiplet Architecture for Ultra-efficient Floating-point Computing", in *Hot Chips: A Symposium on High Performance Chips*, 2020.

[8] Y. S. Shao, et al., "Simba: Scaling deep-learning inference with multi-chip-module-based architecture," in *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture*, 2019, p. 14–27.

[9] L. Song, X. Qian, H. Li and Y. Chen, " PipeLayer: A Pipelined ReRAM-Based Accelerator for Deep Learning," in *IEEE International Symposium on High Performance Computer Architecture (HPCA)*, 2017, pp. 541-552.

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

[11] J. Yin, et al., "Modular routing design for chiplet-based systems," in *ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*, 2018, pp. 726–738.

[12] H. Zheng, K. Wang, and A. Louri, "A versatile and flexible chiplet-based system design for heterogeneous manycore architectures," in *2020 57th ACM/IEEE Design Automation Conference (DAC)*, 2020, pp. 1–6.

[13] A. Shafiee et al., "ISAAC: A Convolutional Neural Network Accelerator with In-Situ Analog Arithmetic in Crossbars," in *ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*, 2016, pp. 14-26.

[14] N. P. Jouppi, et al., "In-datacenter performance analysis of a tensor processing unit," in *ACM/IEEE 44th Annual International Symposium on Computer Architecture (ISCA)*, 2017, pp. 1-12,

[15] C. Chen, et al., "Exploring the Programmability for Deep Learning Processors:from Architecture to Tensorization," in *ACM/IEEE Design Automation Conference (DAC)*, 2018, pp. 1–6.

[16] T. Chen, et al., "TVM: end-to-end compilation stack for deep learning," in *SysML Conference*, 2018.

[17] Y. H. Chen, T. Krishna, J. S. Emer and V. Sze, "Eyeriss: an energy-efficient reconfigurable accelerator for deep convolutional neural networks," in *IEEE Journal of Solid-State Circuits*, vol. 52, no. 1, pp. 127-138, Jan. 2017.

[18] J. Yue et al., "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.