CIMDi: A Chiplet-Based CIM-Digital Heterogeneous DNN Accelerator

***Abstract*—Analog compute-in-memory (CIM) deep neural network (DNN) accelerators can achieve high energy efficiency at the cost of accuracy. Digital DNN accelerators, on the other hand, can handle various dataflows and support high-precision computation, but consume more power than the analog alternatives. This paper proposes CIMDi, a chiplet-based heterogeneous DNN accelerator that integrates both CIM and digital chiplets in the same package. Together with a tailored workload mapping strategy, CIMDi delivers great energy efficiency and guarantees inference accuracy.**

I. Introduction

Deep neural networks (DNNs) achieve high inference accuracy by scaling up their model size [1]–[3]. The increasing complexity of DNN models demands more powerful hardware. Larger DNN accelerators can provide higher computation throughput, but in the meantime consume more energy and are more costly to build due to lower manufacturing yield. The limitation of monolithic chip designs hinder cost- and energy-efficient DNN model deployment on the edge.

Chiplet-based designs integrate multiple small dies into the same package. Leveraging advanced packaging techniques, these small chiplets can communicate and collaboratively deliver high performance that is comparable to a large monolithic accelerator. Compared to large monolithic chips, small chiplets enjoy higher yield and hence are much less expensive to build. Therefore, chiplet designs can potentially enable cost-effective edge deployment. However, pure digital DNN accelerator designs are hitting their memory- and power-wall. Edge devices typically have a stringent energy budget, while both data movement and computation on digital chips consume considerable amount of energy. As a non-von-Neumann architecture, CIM perform computation tasks in memory. Since the computing component of CIM is an analog circuit, the computing speed and energy efficiency are greatly improved compared to that of a digital chip.

The combination of chiplet technology and heterogeneous CIM-digital integration offers new opportunities for designing energy-efficient DNN accelerators. However, chiplet-based designs and CIM architectures do have their own limitations. Firstly, DNN workload mapping in a chiplet-based system is challenging, because inter-chiplet communication complexity and overhead must be considered. Secondly, improving the system-wide resource utilization is challenging, because digital and CIM chiplets have different computation throughputs and operate independently. Thirdly, maintaining the desired inference accuracy is challenging, because CIM architectures typically are implemented in fixed (and low) precision.

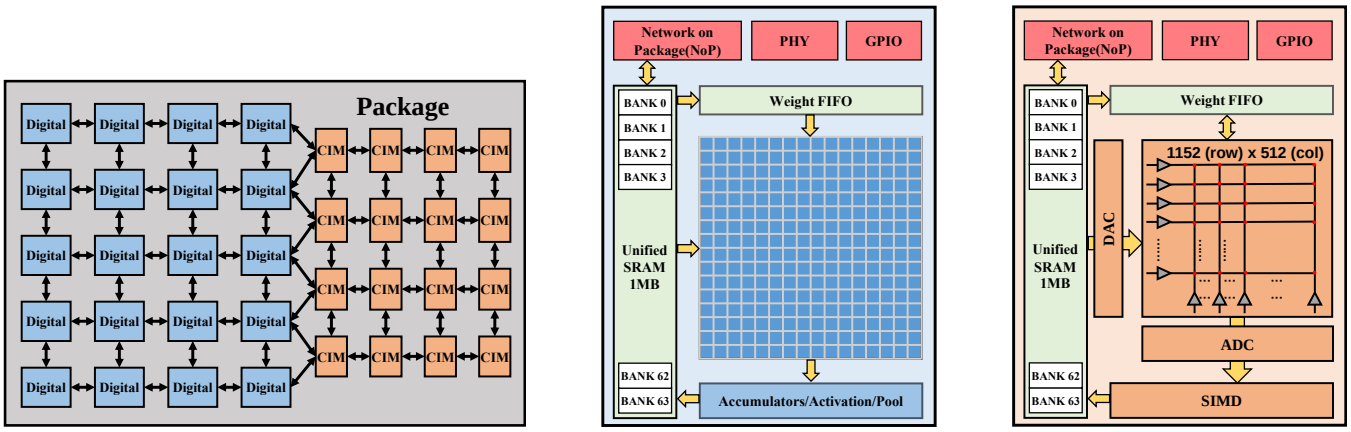
Prior work have looked into chiplet-based digital DNN accelerators [4], [5] and hybrid CIM-Digital system-on-chips [6], [7] separately, but not both in the same design. In this work, we take advantage of both techniques and propose CIMDi, a chiplet-based heterogeneous CIM-digital DNN accelerator. CIMDi is a 36-chiplet package consisting of 20 digital chiplets and 16 CIM chiplets. CIM chiplets can improve the overall computing performance and energy efficiency compared with pure digital designs. The digital chiplets can flexibly handle diverse DNN workloads and support high-precision computation, which makes up for the shortcoming of CIM's fixed and low precision. To exploit the full potential of the hardware, a workload mapping strategy and a pipeline strategy are proposed, tailored for CIMDi architecture. According to the precision requirements of different workloads, CIM or digital chiplets are carefully allocated to improve the overall energy efficiency and resource utilization, meanwhile still maintains reasonable inference accuracy. Compared to state-of-the-are accelerator designs, CIMDi achieves better energy efficiency, demonstrating its capability of balancing energy efficiency and computation precision.

II. Background

A. DNN Dataflow

A DNN workload can be expressed as a sequence of nested loops. The inner-most loop is a set of multiply-and-accumulate (MAC) operations that takes an input activation tensor and a weight tensor, and generates an output activation tensor. Dataflow defines how the DNN workload is partitioned and mapped across different execution units. Based on the dataflow, designers can optimize the hardware and scheduling strategies to explore spatial reuse (e.g., multicasting the same operands across PEs in a timestep) and temporal reuse (e.g., storing the same weight inside a PE and reuse it across time), so as to improve the computation efficiency of DNN accelerators.

Output stationary (OS) and weight stationary (WS) are two types of dataflow that have been widely explored in prior work. In an OS dataflow, input activation and weight values are transmitted across computation units (e.g., PEs), while output values stay inside the same computation units until the computation finishes. In a WS dataflow, weight values are stored inside the same computation units, while inputs are multicasted along each row and outputs are accumulated along each column.



(a) Overview (b) Digital Chiplet (c) CIM Chiplet

Fig. 1: CIMDi Architecture.

B. Spatial Architectures

Digital accelerators. Digital accelerators are flexible for any DNN layer types (convolusion, fully connected, etc.) and dataflows. Typical digital accelerators have structures similar to Figure 1b, where scratchpad memory is partioned into banks to store the operands read from the off-chip DRAM, and PEs are organized in a 2-dimensional matrix to perform MAC operations. Each PE can perform one MAC operation in a cycle, and can support medium to high precision at the expense of higher hardware and energy cost.

CIM accelerators. Compared to digital accelerators, CIM accelerators can achieve better energy efficiency and throughput per unit area at the cost of reduced flexibility and computational precision. Instead of digital PEs, an CIM relies on analog memristor crossbars to perform multiplication and accumulation operations. Typically, CIM accelerators leverages WS dataflow to maximize the temporal reuse of weights. As shown in Figure 1c, digital to analog converters (DAC) and analog to digital converters (ADC) are needed to convert the digital input to analog signals and back to digital when computation is completed. The single-instruction-multiple-data (SIMD) unit is a digital component responsible for post-processing the data and store it back to the scratchpad memory.

C. Chiplet and Advanced Packaging

Chiplets can be integrated into the same package on an organic substrate or an (active or passive) silicon interposer. Chiplets interconnect with each other via serial (e.g., USR, XSR, GRS) or parallel (e.g., BoW, AIB) die-to-die interfaces. Enabled by advanced packaging techniques, the network-on-package (NoP) can provide over 100GB/s/mm inter-chiplet communication bandwidth at 0.25pJ/bit [8]. Compared to large monolithic accelerator designs, chiplet-based systems enjoys higher fabrication yield and lower manufacturing cost [9]. Moreover, heterogeneous chiplet in terms of both functionality and process nodes can be mixed-and-match to improve the system's performance and energy efficiency [10].

III. CIMDi Architecture

Figure 1a is an overview of CIMDi architecture. The 36-chiplet heterogeneous accelerator consists of 20 digital chiplets and 16 CIM chiplets. Chiplets are interconnected through a mesh-like NoP. As mentioned in Section II, digital chiplets can be flexibly reconfigured to support different DNN layers and dataflows. In CIMDi, the digital chiplets support 4-8bit precision computation. Both convolutional and fully connected layers can be mapped to digital chiplets. The CIM chiplets implement a ternary-weight, 7bit-activation low-precision structure. Only DNN layers that are less sensitive to computation precision can be mapped to CIM chiplets.

Digital Chiplet. Figure 1b shows the architecture of a digital chiplet. A digital chiplet consists of a PE array, an accumulator, an activation and pooling unit, a unified SRAM, a weight FIFO, and a NoP router for on- and off-package communication. The PE array contains 256 PE arranged in 16 rows and 16 columns. Each PE has four $4 \times 4bit$ multipliers can perform MAC operations in $4 \times 4bit$, $4 \times 8bit$, and $8 \times 8bit$ precision. The scratchpad memory (denoted as Unified SRAM in the figure) is separated into 64 banks, where each bank is 16KB in size. The 64 banks can be flexibly partitioned to store the input, weight, or output separately, depending on each layer's data size, similar to those proposed in prior works [11], [12]. Activation and pooling operations can be applied to the outputs of the PE array. The resulting data is then stored to the unified SRAM. The NoP router is responsible for transferring data between a pair of chiplets and between chiplet and off-chip memory.

CIM Chiplet. Figure 1c shows the architecture of a CIM chiplet. A CIM chiplet consists of a ReRAM-based crossbar, DAC/ADC, a unified SRAM, a weight FIFO, and a NoP router. The crossbar has 1152 rows and 512 columns, where each cell stores a 2-bit weight. The weight FIFO is used for loading and overwriting the weights stored in the crossbar. The crossbar receives 7-bit inputs from the unified SRAM and generates 6-bit outputs. Since CIM is an analog device, DACs convert digital inputs to analog signals and ADCs transform the signals

back to digital. The SIMD unit will then accumulate the output values, apply activation, quantization, or pooling operations to the output data and store the results back to the unified SRAM.

Network on Package. The NoP adopts a irregular mesh topology, where 20 digital chiplets are organized into a 5-by-4 mesh, and 16 CIM chiplets are organized into a 4-by-4 mesh. As will be discussed in Section IV, there is frequent data transfer between the digital and CIM chiplets. Therefore, we provide more links between the center digital and CIM chiplets, aiming at maximizing the bi-section bandwidth for the CIMDi package.

IV. Computation and Dataflow

Dataflow defines how computation is partitioned and mapped across different chiplets, which will impact the system's performance and energy efficiency. In this section, we introduce our workload mapping and layer pipelining strategies designated for the CIMDi architecture, aiming at maximizing CIMDi's hardware utilization so as to improve the overall energy efficiency.

A. Workload Mapping

We consider each DNN layer an individual workload. In the following discussion, we use layer and workload interchangeably. The workload mapping process determines: 1) whether the workload should be mapped to digital chiplets or CIM chiplets or both, 2) how to partition the workload given the in-package resource, and 3) which chiplet(s) a workload partition should be executed on.

1) Selecting Chiplet Types: The goal of CIMDi is to improve energy-efficiency without significant loss of accuracy. Digital chiplets are flexible and can support high precision, whereas CIM chiplets are energy-efficient and provide high throughput. Workload mapping should take advantage of both. In our design, mapping a workload to digital or CIM chiplets is determined by the workload's sensitivity to computation precision. In other words, if lowering a DNN layer's computation precision has little impact on overall prediction accuracy, we tend to map the layer to CIM chiplets. Otherwise, we map it to digital chiplets, or use both CIM and digital chiplets simultaneously.

Specifically, we introduce two thresholds $Sens_{high}$ and $Sens_{low}$ to classify each DNN layer. After lowering the precision of a layer L, if the overall inference accuracy loss is smaller than $Sens_{low}$ (in this work we empirically set it to 2.5%), then layer L is considered low-sensitive to precision and will be mapped to CIM chiplets. If the accuracy loss is greater than $Sens_{high}$ (we empirically set it to 5%), then the layer is considered high-sensitive to precision and will be mapped to digital chiplets. If the accuracy loss is between $Sens_{low}$ and $Sens_{high}$, then the layer is considered medium-sensitive to precision and can be mapped to both digital and CIM chiplets.

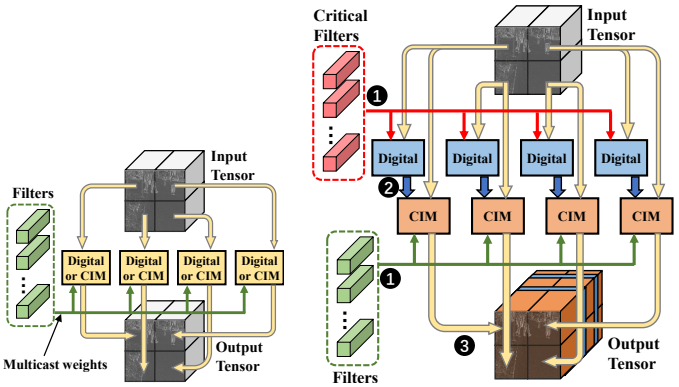
Algorithm 1 Workload Mapping

```
MD_i: minimum number of digital chiplet needed by layer_i
MC_i: minimum number of CIM chiplet needed by layer_i
S_input, S_output, S_filter, S_SRAM: size of input tensor, output tensor, filters, and unified SRAM, respectively

for all layer_i in L do
    if accuracy loss > Sens_high then
        sensitivity = HIGH;
    else if accuracy loss < Sens_low then
        sensitivity = LOW;
    else
        sensitivity = MEDIUM;
    end if
    if sensitivity == HIGH or MEDIUM then
        MD_i = (S_input + S_output + S_filter)/S_SRAM;
    end if
    if sensitivity == LOW or MEDIUM then
        MC_i = (S_input + S_output)/S_SRAM;
    end if
    // layer_i needs at least (m_i MD_i, n_i MC_i) chiplets,
    // where m_i and n_i are integers greater than 1, and m_i MD_i ≤ 20, n_i MC_i ≤ 16
end for
K = [m_0, n_0, m_1, n_1, ..., m_L, n_L];
for all possible combinations of K do
    calculate computation latency T_i in each layer_i
    T = max(T_0, T_1, T_2, ..., T_L);
    if T_0 ≈ T_1 ≈ T_2 ≈ ... ≈ T_L then
        K is taken into consideration
    end if
end for
select a mapping K with minimum T
```

2) Workload Partition: An important feature of CIMDi is that both digital and CIM chiplets can simultaneously compute the same workload in a mixed-precision manner, which improves the energy efficiency while still guarantees the overall inference accuracy. Since in our design, both digital and CIM chiplets adopt WS dataflow, filters that are critical to the inference accuracy are dispatched to digital chiplets, while the rest are sent to CIM chiplets. To determine which filters are critical, we first compute the mean across all filter weights in a given layer, then we rank the filters by the variance of their weight values. The top-most $N\%$ filters that have larger weight variance are candidates for high-precision computation, and should be dispatched to digital chiplets as long as resource allows. The intuition behind is that $8bit$ (used by digital chiplets) to $2bit$ (used by CIM chiplets) weight quantization introduces error. Filters with large weight variance values tend to have a wide weight distribution, which are prone to error during quantization. Note that the exact value of N varies depending on the DNN model and hardware configurations. If N is too large, inference accuracy will improve but fewer filters will be dispatched to the CIM chiplets, which negatively impacts the overall energy efficiency. If N is too small, fewer filters will be dispatched to the digital chiplets and the model might still suffer from high accuracy loss. In our design, after making careful trade-off between accuracy and energy-efficiency, we find 6.25% (i.e., $\frac{1}{16}$ of a layer's filters) a reasonable number for our evaluated workloads.

Algorithm 1 is a brute force algorithm that we use to determine how many digital and/or CIM chiplets should be assigned to each DNN layer. The overall goal of this algorithm is to balance the computation latency across all layers meanwhile minimize the latency of the longest layer, to ensure model



(a) Uniform precision dataflow (b) Mixed precision dataflow

Fig. 2: CIMDi Computation Dataflows.

pipelining. Specifically, lines 6-12 determine the computation precision sensitivity for each layer based on accuracy loss. Lines 13-18 compute the minimum number of digital and/or CIM chiplets required by each layer, based on on-chip storage demands. MD_i and MC_i are minimum number of chiplets because we can potentially provide more chiplets and further tiling the input tensor to speedup the computation. K is one viable workload mapping approach. By varying the values of m_i and n_i, we can iterate through all possible mappings. Lines 23-29 compute the latency for each layer given the mapping strategy. And line 30 finally selects a mapping in which the longest layer has the minimum value while computation latency across all layers is evenly distributed.

3) CIMDi Mixed Precision Computation: Once the number of digital and CIM chiplets for a given layer is determined, filters can be dispatched to the chiplets. In a convolutional layer, the number of output channels equals to the number of filters. If both digital and CIM chiplets are involved in the convolution, output channels will be distributed among both types of chiplets. Therefore, we need a mechanism to gather and reorder the outputs to ensure that data sent to the next layer are in the correct order.

Figure 2 shows how uniform precision and mixed precision computation are handled by CIMDi. For uniform precision computation on either digital or CIM chiplets, the same set of filters is multicasted across the chiplets. Each chiplet receives a tile of the input tensor and generates a tile of the output tensor. For mixed precision computation using both types of chiplet, we introduce a masking mechanism to merge the results. As shown in Figure 2b, critical filters (discussed in Section IV-A2) are multicasted across the digital chiplets (❶). So the digital chiplets are only responsible for computing output channels corresponding to those critical filters. The CIM chiplets, however, will receive ALL filters belonging to a DNN layer (❶). Along with the filters, a mask indicating critical filters is also multicasted to the CIM chiplets. The mask is essentially a sequence of 0s and 1s generated off-line when determining the critical filters, in which '0' indicates a non-critical and '1' indicates a critical filter. When a digital chiplet finishes computation, the results will be sent to the

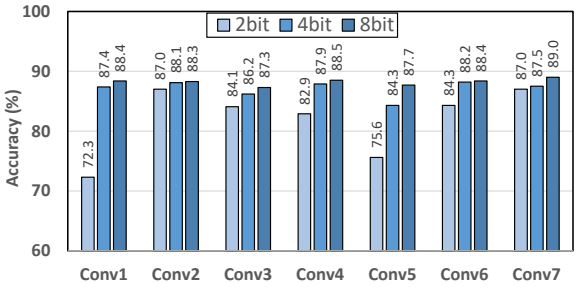
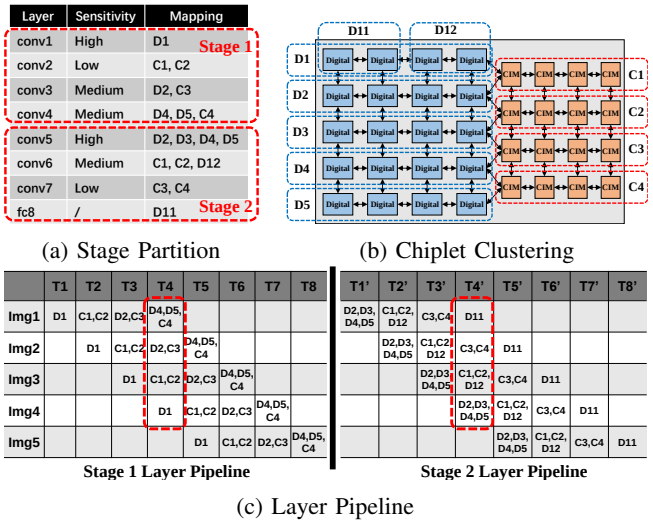


Fig. 3: Precision sensitivity for VGG-8 convolutional layers.



(a) Stage Partition (b) Chiplet Clustering

(c) Layer Pipeline

Fig. 4: Layer Pipelining Example.

CIM chiplet that computes the same tile (❷). Then the CIM chiplet will merge the results in its SRAM buffer according to the mask (❸).

B. Layer Pipelining

In Section IV-A, we introduced our workload mapping strategy that maximizes the utilization of both types of chiplets. In reality, a DNN model can be too large to fit in the accelerator hardware. As a result, intermediate data must be transferred to the off-chip DRAM and later loaded back to the accelerator, which can potentially cause the accelerator to stall and harm the performance. To avoid frequent stalling caused by off-chip data transfer, we propose a layer pipelining strategy to further harvest the potential of CIM-digital heterogeneity. At a high level, we first determine the number of digital and CIM chiplet required for each layer using Algorithm 1, then divide the entire DNN model into multiple stages according to the in-package storage provided by CIMDi chiplets, next we group all digital and CIM chiplets into clusters, lastly we map workloads belonging to the same stage onto the clusters.

Take VGG-8/CIFAR10 as an example. Figure 3 shows the Top-1 inference accuracy under different computation precisions. We can observe that Conv1 and Conv5 are highly precision-sensitive ($2bit$ vs. $4/8bit$) and should be executed on digital chiplets, while Conv2 and Conv7 are precision insensitive and can be executed on CIM chiplets. Conv3, Conv4,

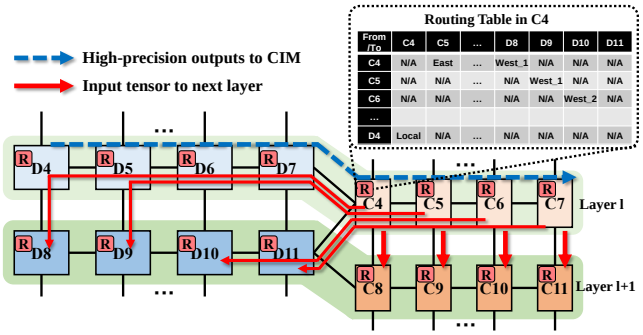


Fig. 5: Network on package routing design.

and Conv6 are medium sensitive, so they can be handled in a mixed precision manner. The workload mapping algorithm generates a mapping strategy and accordingly we can partition the chiplets into clusters (Figure 4b). For example, the Conv1 layer needs four digital chiplets, and we group the top-most four digital chiplets into cluster D1, Conv1 can be mapped to D1 for computation. Considering the computation and storage resource required by each workload and the resource provided by CIMDi, we divide the model into two stages, as shown in Figure 4a. Next, we allocate chiplet clusters to individual workload to maximize the utilization of CIMDi chiplets. As demonstrated in Figure 4c, at a given time, all chiplets are fully utilized during the execution of either stage (circled in red boxes).

C. On-package Communication

Chilet-based designs introduce additional complexity to the on-package network. The NoP must provide sufficient bandwidth and be able to handle diverse communication patterns flexibly. In CIMDi, each chiplet contains a NoP router. Routing tables are used to route on- and off-package traffic. Since the dataflow is determined for each workload, we can pre-configure the routing tables to balance the on-package traffic. As Figure 5 shows, the high-precision outputs generated by digital chiplets flow from west to east, while the merged output tensor (i.e., inputs to the next layer) flows from east to west and north to south. We can program the routing tables to utilize as many on-package links as possible. For example, traffic from C4/C5 to D8/D9 can utilize one of the west links from C4 (denoted as west_1 in the routing table), while traffic from C6/C7 to D10/D11 can take the other west link (denoted as west_2). CIMDi applies an irregular-mesh topology, so routing deadlocks might occur and must be addressed properly. To avoid deadlocks, designers need to analyze the resource dependence graph and break all cyclic dependencies when computing the routing tables.

V. Evaluation

A. Implementation

The proposed CIMDi digital chiplet is implemented in Verilog RTL. We synthesized, placed, and routed the RTL code by Synopsys Development Kits [13] under 28nm CMOS technology [14]. We use CACTI [15] and SRAM Memory Compiler [14] to model the DRAM memory and on-chip SRAM buffers, respectively. In addition, we modeled the CIM chiplet, obtained the performance, area, and power consumption numbers using MNSIM CIM simulator [16]. We employ the ground-referenced-signaling (GRS) [17] as the chiplet-to-chiplet communication interface model that provides a bandwidth of 100 GB/s and consumes 1.17 pJ/bit of energy. The specifications of the digital chiplet and CIM chiplet are shown in Table I and Table II, respectively. The digital chiplet consumes 8.64mm^2 of area, and the power consumption is 868.2mW at the frequency of 600MHz. The area of a CIM chiplet is 6.1mm^2, and the power consumption is 688.4mW at the frequency of 260MHz.

TABLE I: The specification of Digital chiplet.

Technology	28nm CMOS
Die area	2.88mm×3.0mm
Voltage	0.6∼1.0 V
Frequency	50 - 600 MHz
On-chip SRAM	1024 KB
Power	868.2 mW

TABLE II: The specification of CIM chiplet.

Technology	28nm CMOS
Die area	2.42mm×2.5mm
Crossbar Size	1152 × 512
Voltage	0.53∼1.0 V
Frequency	50 - 260 MHz
On-chip SRAM	1024 KB
Power	688.4 mW

B. Experimental Results

1) Benchmark: We use VGG-8 [1] as the benchmark for evaluation. The VGG-8 model includes 7 convolution layers and a fully connected layer. We trained the VGG-8 model under FP32 precision with CIFAR10 [18] dataset on an NVIDIA RTX-A4000 GPU platform [19]. The trained model obtains 89.64% Top-1 inference accuracy.

2) Accuracy and Latency of the Implementation: We execute the trained DNN model VGG-8 and compare the latency and accuracy loss under different hardware configurations. We have designed 4 comparative experiments: i) VGG-8 deployed on NVIDIA GPU; ii) VGG-8 deployed on a multi-chiplet system consisting of 36 digital chiplets under $8\times8bit$ precision; iii) VGG-8 deployed on a multi-chiplet system consisting of 36 digital chiplets under $4\times4bit$ precision; and iv) VGG-8 deployed on a multi-chiplet system consisting of 36 CIM chiplets under $2bit$ precision.

The experimental results are shown in Table III. Compared with the GPU system, our heterogeneous system CIMDi reduces the inference latency by 140× and with only 1.8% accuracy loss. Compared with the system with 36 digital chiplets under $8\times8bit$ precision and the system under $4\times4bit$ precision, CIMDi reduces the inference latency by 6× and 2.1×, respectively. Because of the precision defect of CIM computing, the system with 36 CIM chiplets achieves only 76.32% inference accuracy. Compared with the CIM system, CIMDi has only 0.13ms more inference latency but with negligible accuracy loss.

TABLE III: Accuracy and Latency of the inference

Reconfiguration	Precision	latency (ms)	Top-1 accuracy
RTX-A4000 GPU	32 fp	30.81	89.64 %
36 Digital Chiplets	8×8 bits	1.32	89.42 %
36 Digital Chiplets	4×4 bits	0.46	89.10 %
36 CIM Chiplets	2×7 bits	0.09	76.32 %
CIMDi: 20 Digital Chiplets 16 CIM Chiplets	4×4 bits (Digital) 2×7 bits (CIM)	0.22	87.83 %

C. Comparison

We compare CIMDi against state-of-the-art ASIC designs. The comparison results of performance, power, and energy efficiency are summarized in Table IV. We can see that CIMDi achieves higher inference accuracy than the CIM designs and better energy efficiency than the pure digital architectures. Compare to the well-known MCM design Simba [4], CIMDi achieves 2.4× higher performance and 1.5× better energy efficiency. Compare to the CIM design VLSI [20], CIMDi achieves more than 100× higher performance and 1.38× better energy efficiency. Compare to the ASIC design COMB [21], CIMDi achieves 1.05× better energy efficiency and 300× higher performance.

TABLE IV: Comparison with other designs.

Designs	Simba [4]	VLSI [20]	COMB [21]	CIMDi
Year	MICRO2019	VLSI2021	ISSCC2022	2022
Tech.(nm)	16	65	65	28
Area (mm^2)	6	10.05	11.2	8.64(Digital) 6.8(CIM)
Storage (KB)	752	32	170	1024
Max Freq. (MHz)	2001	296	200	600 (Digital) 260 (CIM)
Precision (bits)	8	2/4/8	3	4/8 (Digital) 2 (CIM)
Power (mW)	30-4160	205	/	868.2 (Digital) 688.4 (CIM)
Performance (TOPS)	4.01 (1 Chip) 127.8 (Package)	0.579 (4b)	0.25 (1 chip) 1.0 (Package)	0.86 (Digital) 21.04 (CIM) 306.1 (Package)
Tops/W	6.24	6.54(4b)	8.6	9.03
Scalability	MCM	No	MCM	MCM

VI. Conclusion

Cost and energy efficiency are critical optimization goals when designing edge devices. Taking advantage of chiplet technology, CIMDi features high energy efficiency and flexibility by combining CIM technique with digital architectures. To address the inherent challenges of chiplet-based heterogeneous accelerator designs, we further propose workload mapping and layer pipelining strategies, which largely improves the in-package resource utilization and energy efficiency without significant loss of accuracy. While this work takes an initial step towards designing chiplet-based heterogeneous accelerators, there remain other fruitful areas for research. The topology and placement of CIM and digital chiplets might have an impact on computation efficiency, so there could be an opportunity for co-designing the DNN algorithm and chiplet placement. The digital and/or CIM chiplets itself can also be heterogeneous, which opens up a new dimension for architectural design and optimization.

References

[1] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *Computer Science*, 2014.

[2] V. Sze, Y. H. Chen, T. J. Yang, and J. S. Emer, "Efficient processing of deep neural networks: A tutorial and survey," 2017.

[3] R. Mayer and H. A. Jacobsen, "Scalable deep learning on distributed infrastructures: Challenges, techniques, and tools," *ACM Computing Surveys*, vol. 53, no. 1, pp. 1–37, 2020.

[4] Y. Shao, J. Clemons, R. Venkatesan, B. Zimmer, M. Fojtik, N. Jiang, B. Keller, A. Klinefelter, N. R. Pinckney, and P. Raina, "Simba: Scaling deep-learning inference with multi-chip-module-based architecture," 2019.

[5] Z. Tan, H. Cai, R. Dong, and K. Ma, "NN-Baton: DNN Workload Orchestration and Chiplet Granularity Exploration for Multichip Accelerators," in *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*, 2021, pp. 1013–1026.

[6] R. Xu, Z. Xiang, L. Tao, and X. Jin, "A CIM-Digital Heterogeneous Accelerating System with Analog Interconnection for Neural Networks," *Journal of Physics: Conference Series*, vol. 1646, p. 012159, 09 2020.

[7] P. Houshmand, G. M. Sarda, V. Jain, K. Ueyoshi, I. A. Papistas, M. Shi, Q. Zheng, D. Bhattacharjee, A. Mallik, P. Debacker, D. Verkest, and M. Verhelst, "DIANA: An End-to-End Hybrid DIgital and ANAlog Neural Network SoC for the Edge," *IEEE Journal of Solid-State Circuits*, pp. 1–13, 2022.

[8] UCIe, "UCIe white paper," https://www.uciexpress.org/general-8.

[9] T. Burd, N. Beck, S. White, M. Paraschou, N. Kalyanasundharam, G. Donley, A. Smith, L. Hewitt, and S. Naffziger, "'Zeppelin': An SoC for Multichip Architectures," *IEEE Journal of Solid-State Circuits*, vol. 54, no. 1, pp. 133–143, 2019.

[10] J. Yin, Z. Lin, O. Kayiran, M. Poremba, M. Shoaib Bin Altaf, N. Enright Jerger, and G. H. Loh, "Modular routing design for chiplet-based systems," in *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*, 2018.

[11] B. Zimmer, R. Venkatesan, Y. S. Shao, J. Clemons, M. Fojtik, N. Jiang, B. Keller, A. Klinefelter, N. Pinckney, P. Raina, S. G. Tell, Y. Zhang, W. J. Dally, J. S. Emer, C. T. Gray, S. W. Keckler, and B. Khailany, "A 0.32–128 TOPS, Scalable Multi-Chip-Module-Based Deep Neural Network Inference Accelerator With Ground-Referenced Signaling in 16 nm," *IEEE Journal of Solid-State Circuits*, vol. 55, no. 4, pp. 920–932, 2020.

[12] Y. Li, A. Louri, and A. Karanth, "Scaling deep-learning inference with chiplet-based architecture and photonic interconnects," in *2021 58th ACM/IEEE Design Automation Conference (DAC)*, 2021, pp. 931–936.

[13] Synopsys, "Design compiler," https://www.synopsys.com/Tools/Implementation/RTLSynthesis/DesignCompiler/Pages.

[14] SMIC, "Memory compiler," https://ww.smics.com/en/.

[15] Kahng, Andrew, B., Shafiee, Ali, Muralimanohar, Naveen, Balasubramonian, Rajeev, and Srinivas, "CACTI 7: New Tools for Interconnect Exploration in Innovative Off-Chip Memories," *ACM Transactions on Architecture and Code Optimization*, vol. 14, no. 2, 2017.

[16] L. Xia, B. Li, T. Tang, G. Peng, and H. Yang, "MNSIM: Simulation platform for memristor-based neuromorphic computing system," in *Design, Automation Test in Europe Conference Exhibition*, 2016.

[17] J. M. Wilson, W. J. Turner, J. W. Poulton, B. Zimmer, X. Chen, S. S. Kudva, S. Song, S. G. Tell, N. Nedovic, and W. Zhao, "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," 2018, pp. 276–278.

[18] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," *Handbook of Systemic Autoimmune Diseases*, vol. 1, no. 4, 2009.

[19] NVIDIA, "TITAN RTX-A4000," https://www.nvidia.cn/design-visualization/rtx-a4000/.

[20] R. Guo, H. Li, R. Liu, Z. Zhang, L. Tang, H. Sun, L. Liu, M.-F. Chang, S. Wei, and S. Yin, "A 6.54-to-26.03 TOPS/W Computing-In-Memory RNN Processor using Input Similarity Optimization and Attention-based Context-breaking with Output Speculation," in *2021 Symposium on VLSI Circuits*, 2021, pp. 1–2.

[21] H. Z. et al, "COMB-MCM: Computing-on-Memory-Boundary NN Processor with Bipolar Bitwise Sparsity Optimization for Scalable Multi-Chiplet-Module Edge Machine Learning," in *2022 IEEE International Solid- State Circuits Conference (ISSCC)*, 2022.