Hardware-Software Co-Design of a Collaborative DNN Accelerator for 3D Stacked Memories with Multi-Channel Data

Tom Glint
IIT Gandhinagar, India
tom.issac@iitgn.ac.in

Manu Awasthi
Ashoka University, India
manu.awasthi@ashoka.edu.in

Joycee Mekie
IIT Gandhinagar, India
joycee@iitgn.ac.in

***Abstract*—Hardware accelerators are preferred over general-purpose processors for processing Deep Neural Networks (DNN) as the later suffer from power and memory walls. However, hardware accelerators designed as a separate logic chip from the memory still suffer from memory wall. Processing-in-memory accelerators, which try to overcome this memory wall by developing the compute elements as part of the memory structures, are highly constrained due to the memory manufacturing process. Near-data-processing (NDP) based hardware accelerator design is an alternative paradigm that could combine the benefit of high bandwidth, low access energy of processing-in-memory, and design flexibility of separate logic chip. However, NDP has area, data flow and thermal constraints, hindering high throughput designs. In this work, we propose an HBM3-based NDP accelerator that tackles the constraints of NDP with a hardware-software co-design approach. The proposed design takes only 50% area, delivers a speed-up of 3×, and is about 6× more energy efficient than state-of-the-art NDP hardware accelerator for inferencing workloads such as AlexNet, MobileNet, ResNet, and VGG without loss of accuracy.**

***Index Terms*—Hardware-software co-design. Deep Neural Network Accelerator, HBM3.**

I. Introduction

Modern Deep Neural Networks (DNN) are capable of image classification, object detection, and segmentation with higher accuracy than humans [1]. However, to achieve such high accuracy, the DNNs have to perform hundreds of millions of Multiply Accumulate (MAC) operations [2]. Several times, these DNNs are deployed in embedded and real-time applications with strict latency and energy constraints [1], [3]. Unfortunately, general-purpose computers can not meet these constraints due to power [4] and memory walls [5]. Conventional hardware accelerators such as Eyeriss [6] and Simba [7] and Processing-in-Memory based accelerators such as UpMem [8], Newton AiM [9] and GDDR6-AiM [10] have been proposed to meet these constraints. Conventional hardware accelerators have efficient compute logic which reduces compute power, but they suffer from memory wall owing to large accesses to external main memory, which leads to high inference energy costs. On the other hand, processing-in-memory accelerators that fabricate compute logic on the same die as memory using the DRAM process have energy efficient memory accesses, but low compute throughput, which, in turn, affects DNN inference time [10].

Near Data Processing (NDP) has the potential to combine the best of conventional and processing-in-memory hardware accelerators, as the logic die and memory are 3D stacked. Several DNN accelerators based on Near Data Processing paradigm have been proposed, such as Tetris [11], Neurostream [12], CLU [13], Hydra [14], Nzespa [15] and [16]. These designs implement the DNN accelerator IPs in the logic layer of 3D stacked memories. The earlier proposals use Hybrid Memory Cube (HMC) [17] and/or High Bandwidth Memory (HBM) [18], [19]) which follows the traditional manufacturing processes, but allows for complex and fast designs. The CMOS logic is connected with the 3D stacked DRAM memories using Through-Silicon-Vias (TSVs), which in turn provide high bandwidth, low energy memory access to the logic layer. Among the existing NDP accelerator designs, Tetris has the highest peak throughput of 3.14 TOPS, while consuming only ~ 4 pJ/MAC operation [11]–[15]. However, these accelerator designs are neither maximized to the constraints of 3D stacked memories nor designed from the ground up for the spatial structure of these 3D stacked memories, and rather are extensions of conventional hardware accelerators such as Eyeriss [6], [11], [16]. This leads to poor resource allocation and decreased throughput. Due to the 3D structure of HMC and HBM, accelerator designs in the logic layer are limited to 15 W Thermal Design Power (TDP) [20], 256 GB/s memory bandwidth [17], [18] and up to 50 mm^2 area [11], [18] constraints.

To the best of our knowledge, this is the first work on NDP which co-optimizes the 3D NDP design and the neural networks structure to best utilize available silicon budget at the logic layer and give the best possible throughput and energy efficiency while not violating the constraints of 3D memory.

The main contributions are: (i) We identify that power/thermal constraint primarily limits the design of NDP accelerators over area and bandwidth. (ii) We perform 3D memory constraint-aware architectural exploration to identify architectural design decisions for high throughput, low energy design without loss of accuracy during inference. (iii) We perform detailed modeling of the proposed architecture by implementing it in the logic layer of HBM3. (iv) We show that the proposed Hardware Accelerator (HA) architecture, **named EnX3D**, achieves 3× speedup and is 6× more energy efficient than SOTA NDP-based accelerator when inferencing workloads such as AlexNet [21], MobileNet [1], ResNet [2] and VGG [22]. (iv) A detailed architectural analysis is performed to identify and verify that resources are not over-provisioned. Along with a sensitivity analysis to show the merit of the design on HMC.

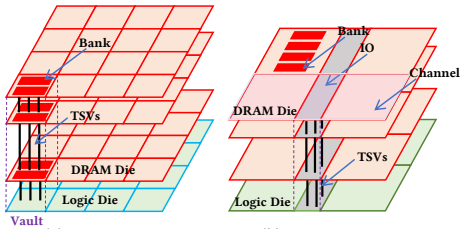
II. Background: 3D Memory

The requirement of high bandwidth, low latency memory has led to the development of 3D memory, where DRAM dies are stacked on top of a logic die. Fig. 1a shows Hybrid Memory Cube (HMC) [17] and Fig. 1b shows High Bandwidth Memory (HBM3) [18], which are two popular instances of 3D memory. In both these instances, the banks of memory in the DRAM dies are connected to the logic die with Through Silicon Vias (TSV). However, the spatial distribution of these TSVs is different in both designs. In HMC, the entire 3D stack

This work is supported through grants received from Science and Engineering Research Board (SERB), Government of India, under SERB-SUPRA grant SPR/2020/000450, and Semiconductor Research Corporation (SRC) through contracts 2020-IR-3005 and 2020-IR-2980

is split into 16 vaults, each having a separate DRAM controller at the logic layer that is connected to the banks directly above using TSVs. TSVs in HBM3 are grouped at the center but provide independent connectivity to sixteen memory channels.

The stacked DRAM dies provide multiple gigabytes of storage capacity. It, therefore, has a large area footprint ($>100mm^2$), and therefore the logic die placed under the DRAM dies share a similar footprint. However, the area inside the logic die is mostly unutilized as the logic die only implements controllers and channels for communication outside the memory package. This unutilized area could be used for implementing a DNN accelerator. Further, HBM has gained a wider adoption than HMC and therefore, this work will also focus on HBM.



(a) HMC (b) HBM3

Fig. 1. Structure of (a) HMC and (b) HBM3 showing the placement of TSVs and Channels/Vaults

A. 3D Memory: Opportunities and Challenges

Capacity: Stacking memory dies as layers provide high density storage solutions with total capacity depending on the total number of layers and manufacturing process. HBM3 implementations upto 24 GB capacities have been demonstrated. This is particularly useful in the case of DNN models, where the size of models can reach several gigabytes.

Bandwidth: For fast processing of DNNs, the Multiply-and-Accumulate (MAC) operations have to be performed in parallel. To perform these parallel operations, the processing elements have to be continuously fed with input and filter data while outputs are stored back from the processing elements. While traditional DIMM-based DDR4 DRAM provides 25 GB/s of bandwidth, HBM3 provides up to 820 GB/s [18], [19]. This allows for more instances of processing elements in the accelerator to be fed with data.

Access Energy: DDR4 interface costs 46 pJ/bit for data transfer while data transfer from the DRAM die to the logic die costs 3.9pJ/bit for HBM3 [19], [23]. However, data transfer to the outside the HBM package requires more than 15pJ/bit.

Flexibility: While the traditional ASIC design flow of DNN accelerators allows for designs with arbitrary dimensions and multiple interfaces, DNN accelerator designs for 3D memory are limited to structural constraints. In the case of HMC, each vault operates under a different clock domain; therefore, designs implemented at each vault must adhere to it. Similarly, HBM3 has 16 banks that provide massive bandwidth, but DNN data has to be partitioned across these banks efficiently so that access can be parallelized.

Thermal Design Point (TDP): The 3D designs are limited to 15 W TDP with low-end active cooling, while with passive cooling, it is further limited to 8.5 W [20].

Area: Due to the organization of the communication channel and other controllers, HMC has a per vault area budget of 3.5 mm^2 while HBM3 is limited $\sim$ 36 mm^2 [16], [18]

III. Experimental Setup

The experimental setup used for a fair and detailed comparison has three significant aspects: (i) The hardware is modeled, and its hardware-bound aspects are captured using a framework made of Timeloop [23], Accelergy [24], Cacti [25], and Aladin [26]. (ii) Both older NNs, like AlexNet [21] and VGG [22], and newer deep NNs, like ResNet [2] and MobileNet [1], are considered as workloads for analysis. (iii) The optimal workload mapping is found for each hardware, rather than a static mapping policy, for a fair comparison of accelerators.

A. Model framework and optimal mapping

Timeloop is used for modeling the DNN accelerator. It captures the spatial organization of a DNN accelerator, including the memory hierarchy (DRAM, SRAM buffers, registers) and the placement of Multiply-Accumulate (MAC) units in relation to the memory hierarchy [23]. It also captures the bandwidth available for communication between each hardware unit. **Accelergy** computes the area and energy associated with sub-components of the DNN accelerator. The energy is calculated on a per-action basis [24]. **Cacti** and **Aladin** are used for finding the area and energy of SRAMs and MAC units. This work uses the 45 nm node for evaluation. Further, the area, latency and energies of compute circuits are obtained from **Cadence Genus**. **Timeloop Mapper** identifies all possible permutations of mapping a workload to the architecture and calculates the latency and energy associated with each mapping based on the Timeloop and Accelergy models. Further, it optimizes for a specified parameter (least delay followed by least latency) and outputs the optimal mapping based on the optimization criteria. *We compare the designs in this work for the optimal mapping case*.

B. Workloads

This work primarily focuses on inference workloads based on CNNs. We use AlexNet and VGG to establish a baseline, as prior works were optimized for them. AlexNet is a DNN with five convolutional layers and has filters with dimensions as large as 11×11. VGG16 is a DNN with substantially more layers and has 13 convolutional layers, and the largest and most common filter size is 3×3. However, we use a more modern and deeper workloads, ResNet and MobileNet from MLPerf inference benchmark suite [1]. In ResNet, we consider only the unique layers, and results from these individual layers can be extrapolated to any ResNet variant.

IV. Design Space Exploration

A. Constraints on the Hardware Accelerator (HA)

Prediction Accuracy: Typically, DNN models are trained with 32-bit IEEE 754 float number system-based computations, which results in very high accuracy [3]. However, the hardware implementation of MAC units that perform 32-bit float point numbers is expensive in terms of latency, energy and area, as seen in Table I. Therefore, modern accelerators quantize the model to fewer bits or to a different number system, such as an integer number system. Furthermore, this quantization usually results in inference accuracy drop. Hence, the data representation choice should have similar accuracy as the base model trained with a float number system.

Latency: DNN applications are typically deployed in scenarios where the input is sourced from a sensor of a sys-

TABLE I
COMPARISON OF CANDIDATE MULTIPLIERS THAT CAN BE IMPLEMENTED AT THE MAC UNIT OF DNN ACCELERATORS, TAKEN FROM [27]

Multiplier	Abbrv.	Type	Bit Width	Bit Arrangment	Latency (ps)	Energy (pJ)	Area (um^2)	LeNet Acc.	ResNet Acc.
IEEE 754 Multiplier(32,8)	FP32	Floating Point	32	(N,es)	1299	22.5	13830.84	98.49%	82.21%
IEEE 754 Multiplier(16,5)	FP16	Floating Point	16	(N,es)	979	4.55	3095.28	98.49%	82.02%
IEEE 754 Multiplier(10,5)	FP10	Floating Point	10	(N,es)	579	1.17	964.08	98.43%	63.81%
Bfloat16 Multiplier(16,8)	BF16	Floating Point	16	(N,es)	832	2.75	2148.12	98.52%	79.19%
Array Multiplier	AM16	Integer	16	-	1780	8.81	6015.24	98.55%	53.18%
Approximate Fixed Posit Multiplier(10,4)	AFPX10	Approximate	8	(N,es)	602	0.607	959.04	98.62%	80.73%

tem, and the system makes decisions based on the result of inference. This is commonly referred to as *Single Stream* scenario [1] and should have low inference latency. MAC operations must be performed with as much parallelism as possible without breaking other constraints to achieve this.

Power: The TDP of 3D memory is limited to 8.5 W with passive cooling and 15 W with low-end active cooling. Hence, the system's overall power should be lower than these limits based on the deployment.

Area and organization: Channels in HBM3 operate independently and fragment the available capacity and bandwidth, making inter-channel communication difficult. Hence a large monolithic accelerator is not viable, and an independently operating distributed system of smaller accelerator instances attached to each channel with very little communication with each other is necessary. Further, due to the overall form factor, the area budget for each instance of the HA attached to a channel can only be less than 2.5 mm^2 for HBM3.

B. Hardware Software Co-Design - Data format of DNN

Table I shows latency, energy, area and accuracy of various multipliers that can be used for inference without retraining the trained model. Traditionally, models are quantized to 8-bit integers or 16-bit integers for low latency and energy, which leads to a significant loss of accuracy for deeper DNNs such as ResNet. 16-bit quantized versions of IEEE float and Brain Float (BFloat) provide one order of magnitude lesser area and energy, but further quantization results in lower inference accuracy, as shown in Table I. [27]–[29] have shown that variants of the posit number system [30], [31] can be used for inference with higher quantization and negligible loss of accuracy compared to 32-bit float. This is possible as this number system encodes data such that numbers near zero value have higher precision, unlike the uniform distribution of precision of IEEE float format, which allows for higher quantization as most of the DNN model data value lies between zero and one [28]. In this work, we choose AFPX10 shown in Table I from [27] due to its low latency, energy, and area while conceding minimal accuracy loss compared to IEEE float format.

C. Collaborative design of HA instance per channel

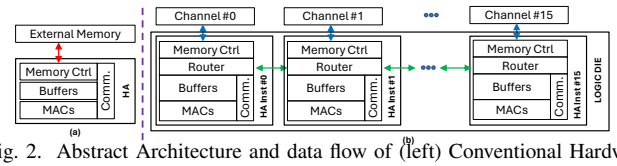


Fig. 2. Abstract Architecture and data flow of (left) Conventional Hardware Accelerator and (right) Hardware Accelerator system for 3D memory based on Near Data Processing Paradigm

The overall data flow in a conventional HA is shown in Fig. 2a where the data flows from a single source. However, in HBM3, the data flows from 16 independent memory channels and to efficiently utilize the available bandwidth and capacity of all the channels, the workload has to split across all channels. We propose using symmetric HA instances that have be designed collaboratively across the channels to efficiently process the split workload. Fig. 2b shows the level 0 data flow diagram of the proposed HA system.

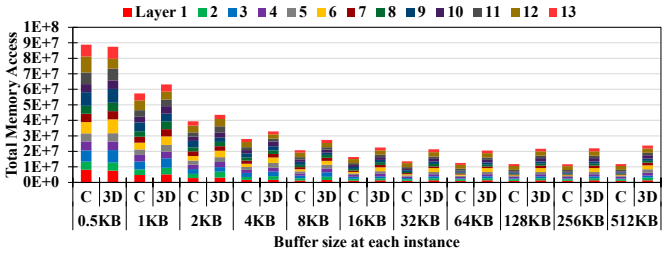


Fig. 3. Access count to DRAM for different buffer sizes for abstract architectures shown in Fig. 2 when inferencing the unique layers of ResNet.

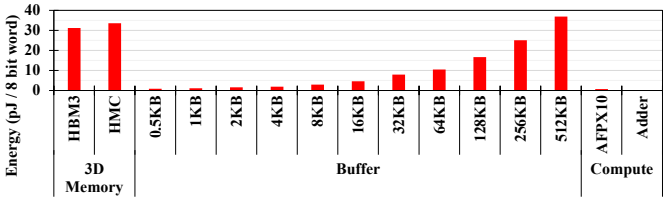


Fig. 4. Energy of components for 8-bit word access or compute

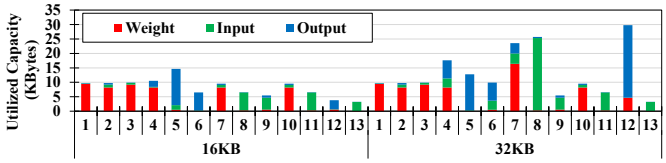


Fig. 5. Utilization pattern of buffer at each HA instance in the abstract architecture shown in Fig. 2b with distributed buffers when inferencing the unique layers of ResNet

Buffers are typically used in HA to cache data to exploit data reuse during processing and reduce the costly accesses to DRAM. To find the smallest buffer capacity required for minimal data access, the abstract architectures specified in Fig. 2 is modeled to find the count of DRAM accesses when processing ResNet. ResNets have layers with inputs with width and height ranging from 224×224 to 7×7 with a variety of channel counts and filter shapes, thus providing a representative workload for identifying organizational requirements of the HA. The resultant access count to DRAM is shown in Fig. 3. Conventional HA requires 256KB of buffer to minimize DRAM access. However, for the proposed design, it is observed that the access count becomes minimal at 64 KB buffer at each channel instance while 16 KB and 32 KB have 3% and 9% more memory accesses due to data duplication. However, it should also be noted that the 16KB buffer consumes 2.2× less energy for access compared to the 64 KB buffer, as shown in Fig. 4. Hence, 16 KB of memory at each HA instance could be the sweet spot considering that only 3 layers out of the 13 unique layers utilize more than 16KB of at each HA instance of the proposed design. However, the total access is more for the distributed system due to data duplication.

Further, as shown in Fig. 5, it is observed that either input or weight only occupies a small capacity at any given time, hence it is further beneficial to split the larger buffer into smaller buffers for inputs, weights and outputs to achieve low access energy. However, as storage requirements for inputs, weights and outputs in the buffer vary drastically from initial layers to deeper layers, another level of buffer storage needs to be introduced. The buffers closer to the Multiply-Accumulation (MAC) unit (one each for input, weight and output) will act as the data source and see the most accesses due to reuse, while a larger global buffer stores the bulk data. Further, groups of these MACs can be organized together as Processing Elements for better resource sharing.

D. Processing Element Architecture

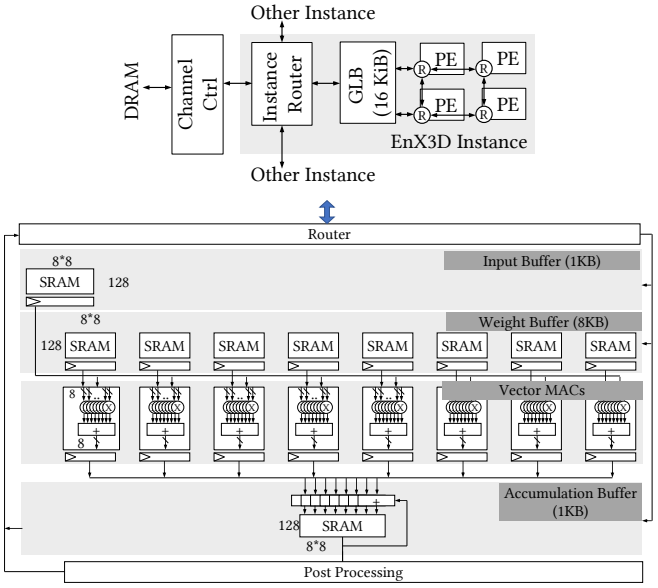


Fig. 6. Top: HA instance at each channel. Bottom: Proposed PE Architecture

Processing Element (PE) organizes a set of MAC units and its associated buffers such that they can be easily controlled and scheduled. Further, the organization allows for shared access to resources within the PE such as buffers. Recent works such as Simba [7] and AiM GDDR6 [10] have shown that the number of buffer access during inference can be reduced by using a set of vector MACs.

The input and output channels of DNNs are typically a multiple of 8 [1], [2], [21], [22]. Hence, a single input word can be shared across eight vector MACs during computation, reducing the number of reads to the buffer storing inputs while ensuring all the Vectors MACs can be utilized for the majority case. Further, within a vector MAC, words from 8 input channels that are on top of each other can be convolved together with the weights belonging to a single filter and accumulated using an adder tree to a single word output, thus reducing writes to the Accumulation/Output Buffer that stores the output data. From Fig. 5, it was observed that the smallest chunk of data that is used repeatedly is between 256 B and 1 KB. Hence the buffers that provide data to the vector MACs are sized to be 1 KB each. In this proposed PE design, as shown in Fig. 6, there are a total of 64 multipliers that can participate in MAC operation during a cycle.

Area and Energy of PE: With the allocation of buffer size and count of Vector MACs as shown in Fig. 6, the area and peak energy consumed by the PE is given in Fig. 7. It can be observed that within the PE, while weight buffers together are the largest area consumer, the Vector MACs consume the most energy.

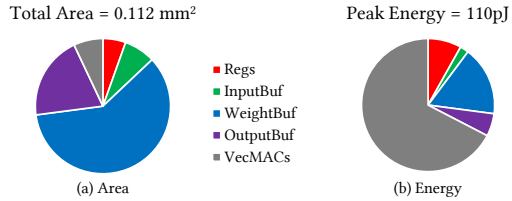


Fig. 7. (a) Area occupied and (b) Peak Energy consumed by a single PE

Architecture of HA Instance: We propose a symmetric HA instance at each channel of HBM3. The PE is taken as an atomic entity that can be connected together using a Network-on-Chip (NoC), which in turn is connected to a Global Buffer (Glb), which can act as the main buffer whose size is identified in Section IV-C. Hence, the total number of PEs should not violate the area and power constraints of the 3D Memory system. When operating at a frequency of 1GHz, each PE consumes a peak power of 0.11 W. With a total TDP of 15 W for the 3D memory, and accounting for the margin for DRAM and Glb access energy, roughly 64 instances of the PE can be instantiated. This translates to 4 PEs per HA instance at the channel level. Further, at the system level, the total area occupied by the design will be 12.37 mm^2, with the GLB occupying 0.1 mm^2 and each HA instance occupying less than 1 mm^2. Hence, the design is limited by TDP, rather than area. The proposed architecture of the HA instance at each channel is shown in Fig. 6.

E. Dynamic Frequency Scaling

With the multiplier being the slowest component in the critical path, based on synthesis of components, the design can operate at a Frequency of up to 1.4GHz, provided the TDP headroom allows it. If the total system power exceeds TDP then the frequency can be brought down by up to 500MHz, albeit costing leakage power.

V. SOTA NDP with highest Peak TOPS

TABLE II
PERFORMANCE AND ENERGY COMPARISON OF STATE-OF-THE-ART ACCELERATORS FROM CHA AND NDP PARADIGMS

INT - Integer, FP - Floating Point, AFPX10 - Approx. Fixed Posit

Paradigm	CHA			NDP			
	STICKER	Exynos	Simba	Neurostream	HYDRA	Tetris	**EnX3D (Proposed)**
Technology (nm)	65	8	16	28	15	45	45
Performance (Peak TOPS)	0.1	1.91	4.01	0.26	3.07	3.14	8.19
Format	8INT	8INT	8INT	32FP	16INT	16INT	AFPOX10
Core Energy (pJ/op/bit)	0.9	0.08	0.52	3.2	2	1.4	0.31
Related Works	[6], [32]–[38]			[11]–[15], [39]–[41], [41]			

Table II shows some relevant related works' details. [11], [13]–[16], [39]–[42] are the relevant DNN accelerator works in the NDP domain. Of these, Tetris [11] (NDP:SOTA) has the highest peak compute throughput of 3.14 TOPS. Further, we note that the performance details of [16] are missing in the literature; therefore, a comparison is not made. Simba [7] (CHA:SOTA) from the CHA paradigm with a peak throughput of 4.01 TOPS and implemented with 16-bit word size is also considered for comparison.

VI. Results and Evaluation

In this section, we examine the performance and constraint metrics of the proposed hardware accelerator EnX3D, implemented on HBM3 and compare it against state-of-the-art conventional and NDP accelerators.

A. Constraint Check

Proposed design should not violate the physical constraints put on it due to the implementation on 3D memory.

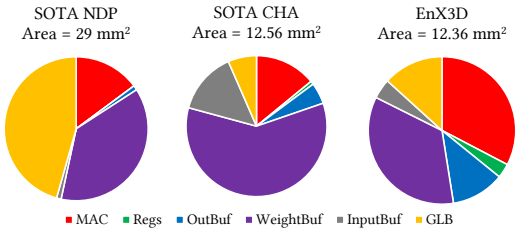


Fig. 8. Area comparison

1) Area

Fig. 8 shows the partition of the area occupied by the complete EnX3D architecture with its 16 instances against SOTA NDP and SOTA CHA accelerators. First, the area of the proposed design is less than 36 mm^2 and therefore does not violate the area constraint of both HMC and HBM3. Second, we note that the area occupied by the global buffer of SOTA NDP is much larger than the proposed design. Third, in the proposed design, the area occupied by the MAC unit is a significantly larger proportion than the other structures, compared to SOTA NDP and CHA designs. Further, compared to SOTA NDP, the area utilization reduces by 2.36×.

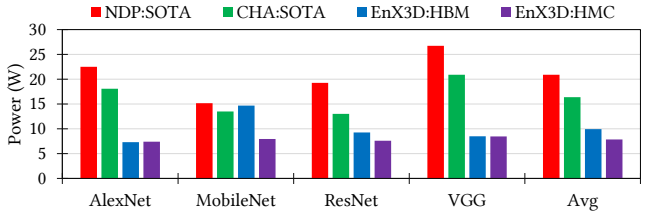


Fig. 9. Thermal Design Power

2) Thermal Design Power (TDP)

Fig. 9 shows the average power drawn by the proposed design when inferencing CNNs. We observe that the power draw of EnX3D:HBM is below the simple active cooling threshold of 15 W.

B. Performance Analysis

HA for DNN workloads needs to have low latency and high energy efficiency. This section compares the speedup and energy efficiency of the proposed HAs.

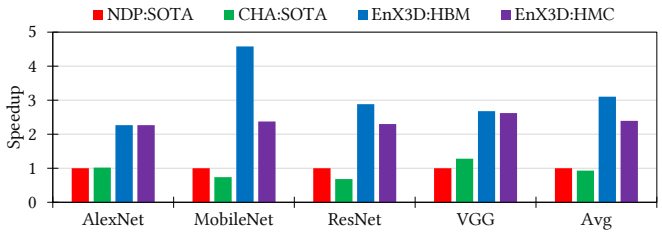


Fig. 10. Speedup

1) Speedup

Fig. 10 shows the speedup of SOTA CHA and EnX3D implementations against SOTA NDP. It is observed that EnX3D:HBM and EnX3D:HMC, on average, has a speedup of 3× and 2.5× compared to SOTA NDP. Further, we observe that for workloads such as MobileNet and ResNet with smaller filter sizes, EnX3D:HBM is significantly faster than EnX3D:HMC. This is because smaller filter size results in fewer data reuse of input words leading to higher bandwidth requirements. HBM3 with 4× the bandwidth of HMC can satisfy the bandwidth requirement, whereas HMC-based implementation is bandwidth bottlenecked. Similarly, SOTA CHA, with 25% higher peak throughput than SOTA NDP, has lower speedup than SOTA NDP for these workloads with smaller filters as the DDR4 DRAM interface has only 10% bandwidth capacity as that of HMC.

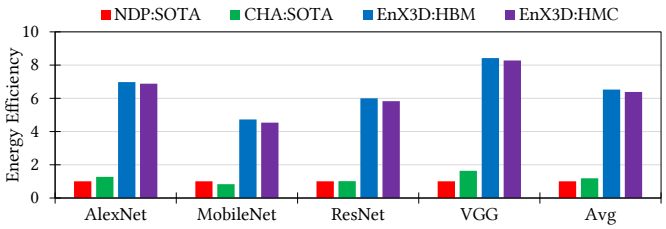


Fig. 11. Energy Efficiency

2) Energy Efficiency

Fig. 11 shows the energy efficiency of the proposed HA compared to SOTA designs as a ratio of energy used for inference for each workload vs baseline NDP. The average energy efficiency of EnX3D is at least 6× higher than NDP:SOTA. Further, we observe that SOTA NDP is more energy efficient for DNNs with smaller filters than SOTA CHA.

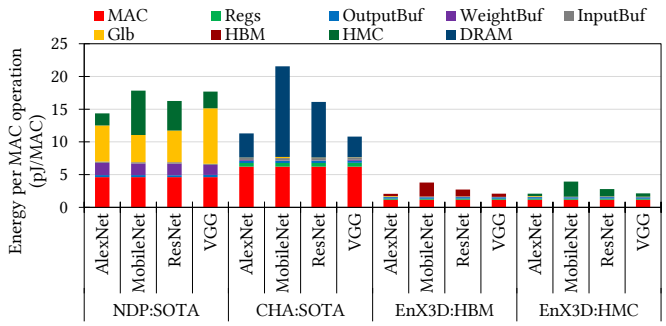


Fig. 12. Amortized energy spent at system level per MAC operation

3) Energy Distribution

Fig. 12 shows the amortized energy spent per MAC operation within the system. We observe that the 16-bit integer MAC unit in CHA:SOTA consumes more energy than NDP:SOTA due to higher operating frequency. However, using the AFPX10 multiplier with posit number system [27], [30], [31], has brought down the MAC energy by ∼5× compared to SOTA HAs. In CHA: SOTA, we observe that the buffer access energy is very low compared to overall energy, but the DRAM access energy dominate and drives the overall energy higher. In NDP:SOTA, the buffer access energy is 48% on average. This can be attributed primarily to two reasons. First, the row stationary approach with individual MACs in the SOTA design leads to a higher access count to the buffers. Second, due to the higher access count, the MAC's local buffers are smaller and cannot hold the necessary data long enough, resulting in re-fetching from the Glb, which in turn increases energy spent on Glb. EnX3D implementations have low buffer access energy due to the Vector MAC, and adder tree design which reduces reads and writes to the local buffers. Further, the 1

KB input buffer, along with the 128 B output buffer, allows the weights read from the weight buffers to be used multiple times before changing to another row of the buffer by rotating the set of inputs present in the input buffer and writing the partial product to different locations in the output buffer. Since the total number of weight words that can be read per cycle is $8\times$ more than input words, this policy saves energy. Further, in EnX3D designs, the global buffer with the size of 16 KB is just sufficient to hold the necessary set of DNN data for computation (as seen in Section IV-C), and the access energy to the Glb is minimal, and the access to the HBM/HMC is also minimal.

VII. Conclusion

Near Data Processing (NDP) paradigm-based accelerators, which implement the compute elements close to the memory location, typically at the logic layer of 3D memories, enjoy high bandwidth and low access energy. However, these designs are physically constrained in terms of area and Thermal Design Point (TDP). In this work, we perform a constraint-aware architectural exploration to find a suitable DNN Hardware Accelerator (HA) design that provides high throughput and low energy without sacrificing inference accuracy. We show that the proposed architecture achieves a speedup of $3\times$ and energy efficiency of $6\times$ compared to the current SOTA NDP-based accelerator with the highest peak throughput. DNNs such as AlexNet, MobileNet, ResNet and VGG are used for this evaluation. The area of the proposed hardware accelerator is less than half of the current SOTA.

References

[1] P. Mattson *et al.*, "Mlperf training benchmark," *arXiv preprint arXiv:1910.01500*, 2019.

[2] K. He *et al.*, "Deep residual learning for image recognition," in *IEEE CVPR*, 2016, pp. 770–778.

[3] Y. Chen *et al.*, "A survey of accelerator architectures for deep neural networks," *Engineering*, vol. 6, no. 3, pp. 264–274, 2020.

[4] X. Guo *et al.*, "Resistive computation: Avoiding the power wall with low-leakage, stt-mram based computing," *ACM SIGARCH computer architecture news*, vol. 38, no. 3, pp. 371–382, 2010.

[5] W. A. Wulf *et al.*, "Hitting the memory wall: Implications of the obvious," *ACM SIGARCH computer architecture news*, vol. 23, no. 1, pp. 20–24, 1995.

[6] Y.-H. Chen *et al.*, "Eyeriss: A spatial architecture for energy-efficient dataflow for convolutional neural networks," *ACM SIGARCH computer architecture news*, vol. 44, no. 3, pp. 367–379, 2016.

[7] 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," *JSSC*, vol. 55, no. 4, pp. 920–932, 2020.

[8] F. Devaux, "The true processing in memory accelerator," in *2019 IEEE Hot Chips 31 Symposium (HCS)*. IEEE Computer Society, 2019, pp. 1–24.

[9] M. He *et al.*, "Newton: A dram-maker's accelerator-in-memory (aim) architecture for machine learning," in *MICRO*. IEEE, 2020, pp. 372–385.

[10] S. Lee *et al.*, "A 1ynm 1.25v 8gb, 16gb/s/pin gddr6-based accelerator-in-memory supporting 1tflops mac operation and various activation functions for deep-learning applications," in *ISSCC*, vol. 65, 2022, pp. 1–3.

[11] M. Gao *et al.*, "Tetris: Scalable and efficient neural network acceleration with 3d memory," in *ASPLOS*, 2017, pp. 751–764.

[12] E. Azarkhish *et al.*, "Neurostream: Scalable and energy efficient deep learning with smart memory cubes," *IEEE Transactions on Parallel and Distributed Systems*, vol. 29, no. 2, pp. 420–434, 2018.

[13] P. Das *et al.*, "Clu: A near-memory accelerator exploiting the parallelism in convolutional neural networks," *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, vol. 17, no. 2, pp. 1–25, 2021.

[14] P. Das *et al.*, "Hydra: A near hybrid memory accelerator for cnn inference," in *DATE*. IEEE, 2022, pp. 1017–1022.

[15] P. Das *et al.*, "Nzespa: A near-3d-memory zero skipping parallel accelerator for cnns," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 40, no. 8, pp. 1573–1585, 2020.

[16] N. Park *et al.*, "High-throughput near-memory processing on cnns with 3d hbm-like memory," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 26, no. 6, pp. 1–20, 2021.

[17] J. Jeddeloh *et al.*, "Hybrid memory cube new dram architecture increases density and performance," in *2012 symposium on VLSI technology (VLSIT)*. IEEE, 2012, pp. 87–88.

[18] J. C. Lee *et al.*, "18.3 a 1.2 v 64gb 8-channel 256gb/s hbm dram with peripheral-base-die architecture and small-swing technique on heavy load interface," in *ISSCC*. IEEE, 2016, pp. 318–319.

[19] M.-J. Park *et al.*, "A 192-gb 12-high 896-gb/s hbm3 dram with a tsv auto-calibration scheme and machine-learning-based layout optimization," *IEEE Journal of Solid-State Circuits*, vol. 58, no. 1, pp. 256–269, 2022.

[20] Y. Eckert *et al.*, "Thermal feasibility of die-stacked processing in memory," 2014.

[21] A. Krizhevsky *et al.*, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, 2012.

[22] K. Simonyan *et al.*, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[23] A. Parashar *et al.*, "Timeloop: A systematic approach to dnn accelerator evaluation," in *ISPASS*. IEEE, 2019, pp. 304–315.

[24] Y. N. Wu *et al.*, "Accelergy: An architecture-level energy estimation methodology for accelerator designs," in *ICCAD*. IEEE, 2019, pp. 1–8.

[25] N. Muralimanohar *et al.*, "Cacti 6.0: A tool to model large caches," *HP laboratories*, vol. 27, p. 28, 2009.

[26] Y. S. Shao *et al.*, "Aladdin: A pre-rtl, power-performance accelerator simulator enabling large design space exploration of customized architectures," in *ISCA*. IEEE, 2014, pp. 97–108.

[27] T. Glint *et al.*, "Hardware-software codesign of dnn accelerators using approximate posit multipliers," in *ASPDAC*, 2023, pp. 469–474.

[28] V. Gohil *et al.*, "Fixed-posit: A floating-point representation for error-resilient applications," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 68, no. 10, pp. 3341–3345, 2021.

[29] J. K. Devnath *et al.*, "A mathematical approach towards quantization of floating point weights in low power neural networks," in *2020 33rd International Conference on VLSI Design and 2020 19th International Conference on Embedded Systems (VLSID)*, 2020, pp. 177–182.

[30] J. Gustafson, "Posit arithmetic," *Mathematica Notebook describing the posit number system*, 2017.

[31] J. L. Gustafson *et al.*, "Beating floating point at its own game: Posit arithmetic," *Supercomputing frontiers and innovations*, vol. 4, no. 2, pp. 71–86, 2017.

[32] M. Alwani *et al.*, "Fused-layer cnn accelerators," in *MICRO*. IEEE, 2016, pp. 1–12.

[33] T. Chen *et al.*, "Diannao: A small-footprint high-throughput accelerator for ubiquitous machine-learning," *ACM SIGARCH Computer Architecture News*, vol. 42, no. 1, pp. 269–284, 2014.

[34] S. Han *et al.*, "Eie: Efficient inference engine on compressed deep neural network," *ACM SIGARCH Computer Architecture News*, vol. 44, no. 3, pp. 243–254, 2016.

[35] A. Parashar *et al.*, "Scnn: An accelerator for compressed-sparse convolutional neural networks," *ACM SIGARCH computer architecture news*, vol. 45, no. 2, pp. 27–40, 2017.

[36] W. Qadeer *et al.*, "Convolution engine: balancing efficiency & flexibility in specialized computing," in *ISCA*, 2013, pp. 24–35.

[37] H. Sharma *et al.*, "Bit fusion: Bit-level dynamically composable architecture for accelerating deep neural network," in *ISCA*. IEEE, 2018, pp. 764–775.

[38] F. Sijstermans, "The nvidia deep learning accelerator," in *Hot Chips*, vol. 30, 2018, pp. 19–21.

[39] S. Yin *et al.*, "Parana: A parallel neural architecture considering thermal problem of 3d stacked memory," *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 1, pp. 146–160, 2018.

[40] Y. Wang *et al.*, "Exploiting parallelism for cnn applications on 3d stacked processing-in-memory architecture," *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 3, pp. 589–600, 2018.

[41] N. Park *et al.*, "High-throughput near-memory processing on cnns with 3d hbm-like memory," *TODAES*, vol. 26, no. 6, pp. 1–20, 2021.

[42] E. Azarkhish *et al.*, "Neurostream: Scalable and energy efficient deep learning with smart memory cubes," *IEEE Transactions on Parallel and Distributed Systems*, vol. 29, no. 2, pp. 420–434, 2017.