Hydra: A near hybrid memory accelerator for CNN inference

Palash Das
Department of CSE
IIT Guwahati, Guwahati, India
palash.das@iitg.ac.in

Ajay Joshi
Department of ECE
Boston University, Boston, USA
joshi@bu.edu

Hemangee K. Kapoor
Department of CSE
IIT Guwahati, Guwahati, India
hemangee@iitg.ac.in

***Abstract*—Convolutional neural network (CNN) accelerators often suffer from limited off-chip memory bandwidth and on-chip capacity constraints. One solution to this problem is near-memory or in-memory processing. Non-volatile memory, such as phase-change memory (PCM), has emerged as a promising DRAM alternative. It is also used in combination with DRAM, forming a hybrid memory. Though near-memory processing (NMP) has been used to accelerate the CNN inference, the feasibility/efficacy of NMP remained unexplored for a hybrid main memory system. Additionally, PCMs are also known to have low write endurance, and therefore, the tremendous amount of writes generated by the accelerators can drastically hamper the longevity of the PCM memory. In this work, we propose Hydra, a near hybrid memory accelerator integrated close to the DRAM to execute inference. The PCM banks store the models that are only read by the memory controller during the inference. For entire forward propagation (inference), the intermediate writes from Hydra are entirely performed to the DRAM, eliminating PCM-writes to enhance PCM lifetime. Unlike the other in-DRAM processing-based works, Hydra does not eliminate any multiplication operations by using binary or ternary neural networks, making it more suitable for the requirement of high accuracy. We also exploit inter- and intra-chip (DRAM chip) parallelism to improve the system's performance. On average, Hydra achieves around 20x performance improvements over the in-DRAM processing-based state-of-the-art works while accelerating the CNN inference.**

***Index Terms*—Convolutional Neural Network, Accelerator, Near-memory Processing, Hybrid Memory**

I. Introduction

Convolutional neural network (CNN) has become the preferred deep learning algorithm in several real-life applications like object detection, image classification, and image segmentation. CNNs primarily comprise of two phases: training and inference. Compared to inference, a model's training is heavier in computations and memory requirements, as millions of parameters are iteratively updated over multiple epochs. However, over the years, the CNN models have evolved into deeper networks with several hidden layers and larger input datasets. Consequently, the inference phase has started having high computations and memory requirements (millions of parameters and up to tens of billions of operations/frame).

Various accelerators have been developed to enhance the performance/energy efficiency of CNN inference. Key limitations faced by many of the CNN accelerators are limited off-chip memory bandwidth or on-chip capacity, leading to *memory-wall* problem. One solution to the problem is near-memory processing (NMP), where computations are offloaded to the processing units, placed near the memory. Inference has been accelerated in prior works using the NMP concept [1], [8], [11]. These works integrate custom hardware near 3D DRAM-based memories for the acceleration of CNNs. Efforts have also been spent to implement traditional DRAM-based accelerators that use processing-in-memory (PIM) for CNNs [4], [14]. However, these works avoid crucial multiplications using binary/ternary neural networks, as implementing multiplication in PIM is challenging. Avoiding multiplications by exploiting quantized networks leads to an energy-efficient solution for mobile/IoT devices. However, there is a non-negligible inference error of around 7% for ImageNet top-1 accuracy, between binary neural network [2] and the best case [25].

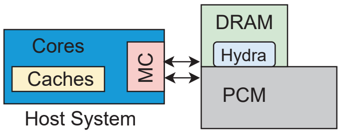


Fig. 1: A high-level view of the proposed system.

Emerging data-intensive applications demand high memory capacity. Limitation in DRAM scaling is unable to meet the growing memory requirement. Non-volatile memories (NVM) such as phase-change memory (PCM) has emerged as a potential candidate for main memory. The NVMs offer high memory density, low cost per bit, and near-zero standby power consumption in exchange for low performance and limited endurance [5], [18]. Though NVMs, like PCM, provide several advantages, they cannot entirely replace the DRAM. Instead, it is more practical to use NVMs in conjunction with DRAM to form a hybrid memory system [5], [18], [24]. The NMP has been extensively explored for accelerating CNNs [1], [4], [8], [11], [14]. However, the efficacy/feasibility of NMP is not investigated for the emerging hybrid main memory while executing the CNNs.

This work aims to build a system with NMP capability in hybrid memory that benefits inference operations in terms of performance and energy efficiency. We choose one of the most promising main memory subsystems; specifically, hybrid memory [5], [15], [23]. Figure 1 presents a conceptual view of the proposed system where we integrate custom hardware, Hydra, closer to DRAM for inference processing. The Hydra

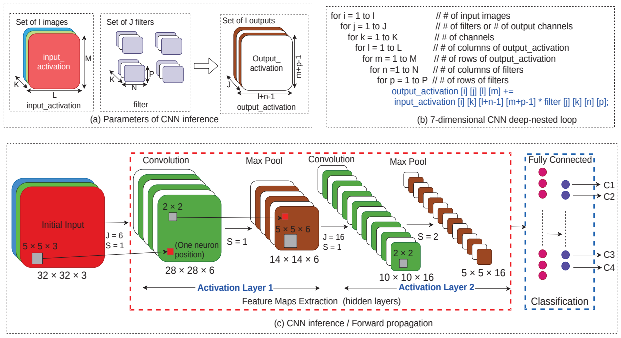
978-3-9819263-6-1/DATE22/©2022 EDAA

units are designed to operate with hybrid main memory as they prefetch model parameters from PCM in parallel to inference execution. We choose the DRAM-PCM combination (shown in Figure 1) among the various DRAM-NVM combinations available in hybrid memory architectures as PCM is one of the most mature candidates for main memory [17]. However, our proposed hardware is memory-technology independent and can be integrated with any DRAM/NVM-based memory. The reason is; unlike [4], [14], we rely entirely on the proposed hardware (Hydra) for all the CNN computations, and we do not modify the DRAM or PCM memory cells for any computation. We use these memory cells only for traditional load/store operations, making the design less complex for practical implementation. Consequently, unlike [4], [14], our architecture is not restricted to quantized binary/ternary models as Hydra has the capability of executing the multiplications. We place the Hydra units close to the memory to reduce the data access cost in terms of access latency and energy consumption without any significant change in the memory sub-arrays.

We design custom hardware, *Hydra, a near hybrid memory accelerator* for CNN inference, and integrate 16 Hydra units (one per chip) inside the DRAM DIMM (dual in-line memory module) in the hybrid memory subsystem. These 16 Hydra modules can work in parallel, and all the multiply-accumulate (MAC) units of each Hydra module can also concurrently execute the inference tasks, leading to inter- and intra-chip parallelism, respectively. We integrate the Hydra module inside DRAM as a design choice in the DRAM-PCM hybrid memory subsystem. This design choice helps in eliminating the intermediate writes of the CNN's hidden layers in the PCM's cell array, leading to an enhanced lifetime for the PCM device [7]. While executing inference close to the DRAM, we choose to keep models in PCM and the data generated during the inference in the DRAM. The reasons for keeping the models in denser PCM are: ❶ Models are usually heavier (in size) than a sample under classification, ❷ In inference, the model's parameters are not updated, and hence no PCM writes are required, ❸ The read latency of both PCM and DRAM is equivalent, and ❹ After the initial prefetching of data, page migration latency is overlapped by the CNN's execution time as we keep buffers in Hydra to store both the input activations and weights. The primary contributions of this work are as follows.

1) We design dedicated hardware, Hydra, and integrate those units inside the DRAM's chips in the hybrid main memory subsystem. This integration helps in alleviating the off-chip memory accesses using the NMP concept.
2) We also employ data partitioning to leverage intra- and inter-chip parallelism inside DRAM for the concurrent execution of the CNN tasks using multiple Hydra units.
3) The Hydra controller prefetches the models from PCM (only read) into its local buffers and overlaps the page migration latency by the execution latency. The Hydra does not use its PCM module for CNN's intermediate write operations and thus enhances the lifetime of PCM.

The proposed system outperforms the state-of-the-art while accelerating inference. Unlike DRAM-based inference accelerators, our architecture does not restrict the execution of non-quantized models. The throughput of the proposed system is evaluated with popular networks like VGG-16 [21], VGG-19 [21], and ResNet-34 [9], pre-trained on ImageNet dataset [3].



(a) Parameters of CNN inference

```
for i = 1 to I                          // # of input images
  for j = 1 to J                        // # of filters or # of output channels
    for k = 1 to K                      // # of channels
      for l = 1 to L                    // # of columns of output_activation
        for m = 1 to M                  // # of rows of output_activation
          for n =1 to N                 // # of columns of filters
            for p = 1 to P              // # of rows of filters
              output_activation [i] [j] [l] [m] +=
               input_activation [i] [k] [l+n-1] [m+p-1] * filter [j] [k] [n] [p];
```

(b) 7-dimensional CNN deep-nested loop

(c) CNN inference / Forward propagation

Fig. 2: An abstract architecture of CNN inference.

II. BACKGROUND

A. CNN fundamentals

The CNN inference commonly includes four layers: Convolution (CONV), Pooling, Non-linear, and Fully-Connected (FC). Figure 2 represents the architecture of a feed-forward inference phase. The CNN algorithm starts its execution by extracting the basic features. The algorithm propagates through the hidden layers for more complicated features in the subsequent layers. The CONV layers are packed with dot products of two matrices (input_activations and weights) at each neuron position (shown in Fig. 2). Pooling layers downsample the feature maps. Output feature maps of one layer are passed as the input feature maps to the next layer. A new set of weight matrices representing more sophisticated features is then applied in the subsequent CONV layers. The inference phase terminates with the prediction results for the set of input samples in the classification layer. The 7-dimensional deep-nested loops (Figure 2b) represent operations involved in each activation layer. Since multiply-add operations are associative in nature, the sequence of loop executions can be reordered. Consequently, the computations can be distributed among the various processing units (Hydras) to achieve parallelism.

B. Hybrid main memory subsystem

Several works on hybrid main memory [5], [15] exploit the benefits of both DRAM and NVM (like PCM). The DRAM-PCM-based hybrid memory architecture can be classified into two categories: (1) DRAM-PCM parallel architecture, and (2) DRAM as a cache for PCM [15]. We consider DRAM-PCM parallel architecture with Hydra integrated into the DRAM chips. The benefits of using a hybrid memory are multi-dimensional. The hybrid main memory subsystem exploits the benefits of both DRAM and PCM (high capacity, low standby power, non-volatility of PCM + high performance, better active power of DRAM) while minimizing the negative impact of both memories [10].

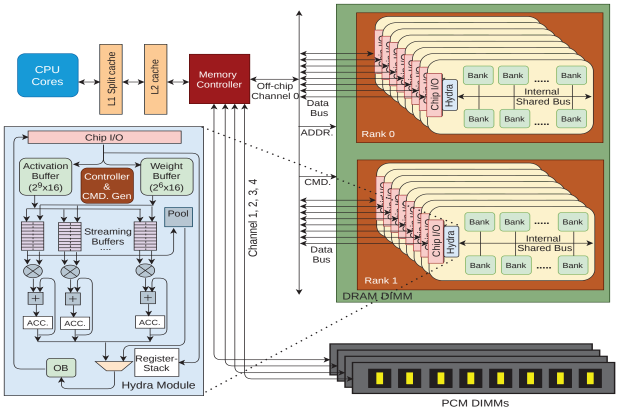


Fig. 3: The proposed system.

Modern DRAMs are typically multi-banked, each serving requests independently from others. Each bank is composed of mats or the 2D-subarray of memory cells. An entire row of data is brought to the row buffer with the help of control wires like wordlines and bitlines. The row buffers primarily include a set of sense amplifiers that amplify the small change of voltage to a stable voltage level. From the row buffer, a few bits are sent to the I/O pads by the column decoder.

PCM is a chalcogenide glass that has large resistance contrast between crystalline and amorphous states [19]. The difference in resistance is used for representing 1 and 0, or even multi-state/cell. The heating process drives the phase transitions between the two states of PCM. Unlike DRAM, PCM does not require refresh operation. Despite all the benefits, the heating processes involved in changing PCM states are long and power-consuming. Consequently, writing operations on PCM is costly in terms of latency and power consumption than a DRAM write. Similar to DRAM, the PCM is also hierarchically organized with channels, ranks, chips, and banks in our proposed system [22]. The detailed parameters of the DRAM-PCM hybrid memory are shown in Table I.

III. Proposed Architecture

A. Overall System Architecture

Figure 3 shows the overall system architecture. The host side includes a quad-core host processor with a two-level cache hierarchy (L1 split and L2 shared). We choose PCIe interface-based DRAM-PCM parallel hybrid memory for the near-memory integration of the proposed Hydra units. Similar to [23], we use one channel for DRAM and the other four channels for PCM memory, as shown in Figure 3. A rank is composed of multiple chips to increase the width of the memory channel. To leverage the chip-level parallelism available via rank, we integrate 16 Hydra units, one per DRAM chip (chip-level integration), inside the DRAM module (shown in Figure 3). This chip-level integration is more area efficient than a bank-level integration, as the number of hardware (Hydra) is less in chip-level integration compared to the bank-level integration. The same has been manifested in [13]. Within each DRAM chip, the Hydra module is connected with the internal bus that is shared by all banks. The CNN inference data, like weights, input activations, are fetched into the local buffers of Hydra modules through the arbitration of the internal shared bus.

B. Hydra Microarchitecture

The abstract micro-architecture of the Hydra unit is also shown in Figure 3 (left blue box). The primary components of a Hydra are Hydra controller and command generator, weight buffer (WB), activation buffer (AB), output buffer (OB), 32 small streaming buffers, 32 fast and efficient MAC units, register-stack, and pool unit with 32 comparators. Additionally, a result accumulation unit with an integrated linear function is also integrated with the memory controller. The host processor initiates the inference process and sends the meta-data information like hyperparameters of the ConvNets to the register-stack with the help of memory controller. Hydra controller drives all the components to accomplish the inference tasks. Each Hydra controller loads one channel of input-maps and kernels into its local AB and WB, respectively, from DRAM through the internal shared bus. To save the command bandwidth, basic command generation can also be done by the Hydra controller, similar to [13]. The filling of AB and WB is done by the WRITE command. The Hydra architecture has been configured to operate on 16-bit fixed-point precision. However, our design can be easily adapted for other precisions. The Hydra controller continuously loads the streaming buffers by the weight-activation, $\langle W, A\rangle$, pairs to be processed by the MAC units. The bias register in register stack holds the bias value that is added only once to each neuron position. After MAC operations, the Hydra controller stores the results in the OB of the Hydra modules. Finally, the results from the OB can also be read by the memory controller using the READ commands for accumulation in the result accumulation unit. Note that we have not shown all the connections in Figure 3 for simplicity and readability.

C. Dataflow

The proposed architecture with integrated Hydra exploits both the intra- and inter-chip parallelism to maximize the system's throughput. Figure 4 represents the data distribution and its granularity of execution. The upper portion of Figure 4 shows the inter-chip parallelism. Each channel of inputs and weights of a hidden layer is sent to one Hydra unit of a chip for the parallel execution of convolution (CONV)/pool operations. For the initial layer, all Hydra units may not get a channel to be processed because of the lesser input/weight channels in the initial layer of CNNs. However, for remaining all hidden layers, a batch of 16 channels can be processed concurrently by 16 Hydra units, one at each chip. Note that we consider a dual-rank, 8 chip/rank, 16 bank/rank configuration of DRAM (2Rx16). Increasing the number of Hydras by increasing the number of chips or ranks can further increase the system's performance due to the additional inter-chip parallelism.

The intra-chip parallelism within one Hydra unit is represented in the lower portion of Figure 4. The 32 MAC units

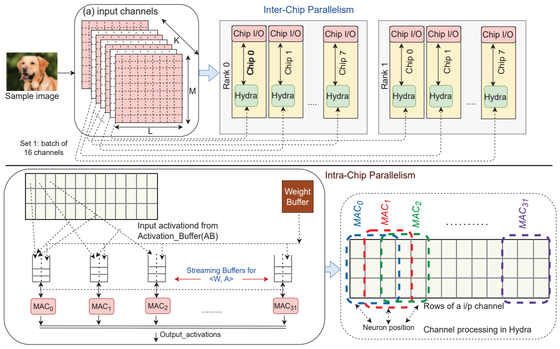


Fig. 4: Dataflow to leverage parallelism.

of a Hydra can perform the CONV/pool operations at 32 neuron positions of a channel in parallel to others, leading to intra-chip parallelism. The Hydra controller schedules the CNN tasks to each MAC, based on the available neuron positions to be processed. Each MAC is associated with a small stream buffer that holds the $\langle W, A\rangle$ pairs to be processed. The Hydra controller always keeps these buffers operation-ready by filling them with the appropriate $\langle W, A\rangle$ pairs to be processed by the corresponding MAC units. The buffer filling time is overlapped with the MAC operations. The Hydra controller reduces the number of fetch operations by fetching data once and broadcasting them to all the appropriate streaming buffers during the filling of the streaming buffers with the $\langle W, A\rangle$ pairs, as shown in Figure 4. The model parameters can be prefetched from PCM banks through the data bus and memory controller in parallel to inference tasks. The combination of intra- and inter-chip parallelism enables Hydra to achieve high performance.

TABLE I: Specification of the architectures

Host Processor (Common for all systems)	
Core	x86-64, 4 core, 4 GHz
Cache Memory (Common for all systems)	
L1 i-cache	32 KB, private, 4 way, 64 B blocks
L1 d-cache	32 KB, private, 8 way, 64 B blocks
L2 cache	256 KB, shared, 8 way, 64 B blocks
DRAM	
Timings [12]	t_{RP} = 5 cy, t_{CCD} = 4 cy, t_{RCD} = 5 cy, t_{CL} = 5 cy, t_{WL} = 4 cy, t_{WR} = 6 cy, t_{RTP} = 3 cy
Energy [12]	1.17 pJ/bit (array read), 0.39 pJ/bit (array write) 0.93 pJ/bit (buffer read), 1.02 pJ/bit (buffer write)
Configuration	8GB, 1 channel, 2 rank, 8 chip, 16 banks, FR-FCFS request scheduling
PCM	
Timings [12]	t_{RP} = 60 cy, t_{CCD} = 4 cy, t_{RCD} = 22 cy, t_{CL} = 5 cy, t_{WL} = 4 cy, t_{WR} = 6 cy, t_{RTP} = 3 cy
Energy [12]	2.47 pJ/bit (array read), 16.82 pJ/bit (array write) 0.93 pJ/bit (buffer read), 1.02 pJ/bit (buffer write)
Configuration	32GB, 4 channel, 8 rank, 8 banks/rank, FR-FCFS request scheduling
Proposed Hydra	
# of Hydra units, # of MACs / Hydra	16, 32
Activation buffer, Weight buffer per PE	1 KB, 128 B
Power, Area (total), Frequency (15 nm CMOS tech.)	1.4 W, 1.6 mm^2, ~3 GHz
Precision	FX16

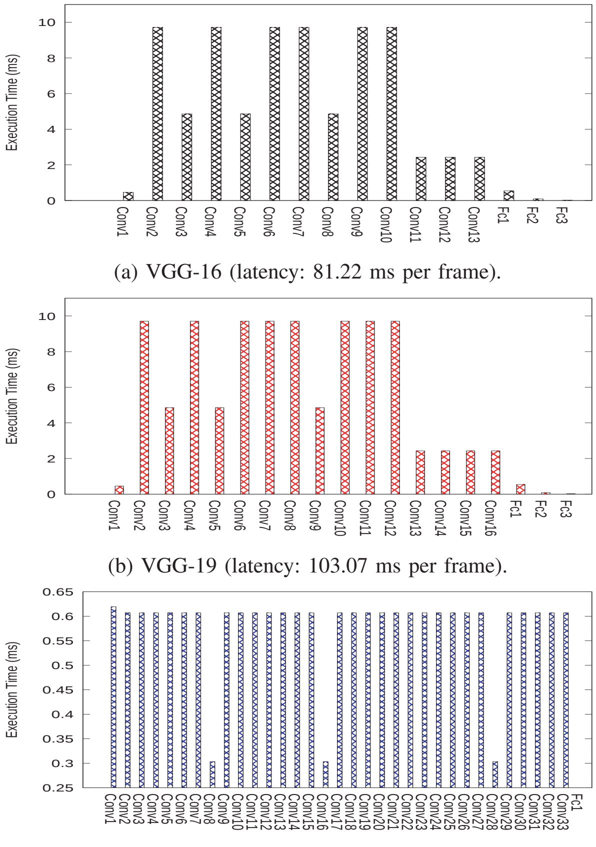
IV. Evaluation

A. Evaluation Methodology

To measure efficacy of the proposed system, we conduct experiments using the popular CNN workloads like VGG-16 [21], VGG-19 [21], and ResNet-34 [9], pre-trained on ImageNet dataset [3]. The data and network parameters are extracted from the experiments in the PyTorch framework [16]. The detailed configuration of the proposed architecture is mentioned in Table I.

B. Results

We evaluate our proposed system with integrated Hydra based on its performance and energy consumption. While incorporating the NMP concept in hybrid main memory, the area of the proposed units is highly critical. Hence, we also perform a detailed area analysis of the Hydra modules in the subsequent section.



(a) VGG-16 (latency: 81.22 ms per frame).

(b) VGG-19 (latency: 103.07 ms per frame).

(c) ResNet-34 (latency: 19.13 ms per frame).

Fig. 5: Performance of CNN inference in Hydra.

1) Performance and Energy analysis: We implement the Hydra module in Verilog hardware description language. We use Genus Synthesis Solution (version 15.21) from Cadence and 15 *nm* CMOS technology for the placement-aware logic synthesis of the designed Hydra. After synthesis, we obtain an operating frequency of around 3 *GHz*, as shown in Table I. We develop an in-house cycle-accurate simulator that resembles the state machines of the designed hardware, similar to [11]. Our simulator is parametrizable and can be fed with the frequency obtained from synthesis. We obtain the network's layer-wise execution times (ms) from the simulator, which include both

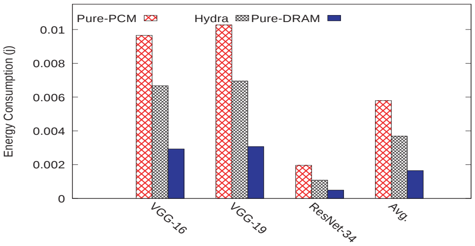


Fig. 6: Comparison of data transfer energy per frame.

data processing and load/store latency. Figure 5 shows the layer-wise latency in milliseconds (ms) for all networks in the Hydra architecture. The total execution latency per frame is also shown in Figure 5 for all the networks. Note that VGG-16 and VGG-19 have a similar trend in the execution latency as they have similar CNN architecture. However, VGG-19 has more Conv layers than VGG-16, leading to an additional latency of 21.85 ms per frame in Hydra. The ResNet-34 takes less time (ms) to process one frame as the number of parameters in ResNet-34 is less compared to VGG-16/VGG-19 network.

For a comparative study on the system's throughput, we also integrate our hardware with a pure DRAM (no PCM attached) and a pure PCM (no DRAM attached) based system. We keep the number of hardware units, place of integration, and dataflow the same for all systems. We obtain an equivalent performance for all systems (pure DRAM, pure PCM, and Hydra) though the memory access latencies are different. The reason is: Hydra is specially designed with additional buffers to overlap data access latency with its execution latency. Additionally, execution latency is the same as we integrate the same hardware for all the systems. Consequently, we obtain equivalent performance for all the systems.

However, we observe a significant impact on the data transfer energy across all three systems. We measure the post-synthesis power consumption of the proposed hardware on Genus from the Cadence. At 15 *nm* CMOS technology, we find the power consumption to be 1.4 W in total for all 16 hardware units. Figure 6 shows the comparison of data transfer energy (joule) for 1 frame in all the systems. On average, Hydra consumes 1.6x lower energy than the pure PCM-based system. However, the pure DRAM-based system stands out to be the best (3.5x lower energy than pure PCM-based system) among these systems. The reason is: the array read, write energies of PCM are respectively 2.1x and 43.13x more than DRAM [12].

2) Write Reduction on PCM: We integrate Hydra modules into DRAM chips. This integration aims to eliminate intermediate writes on the PCM while executing the hidden layers of CNNs. Figure 7 presents the layer-wise estimates of write reduction on the PCM modules for one frame in VGG-16 and VGG-19. A total of *1.49E+07* and *1.36E+07* numbers of writes per frame are avoided in VGG-19 and VGG-16, respectively, for the PCM memory. For one inference epoch on ImageNet test-set [3], the savings on the number of writes are *1.49E+12* and *1.36E+12* for VGG-19 and VGG-16, respectively, leading to increased lifetime for PCM [7].

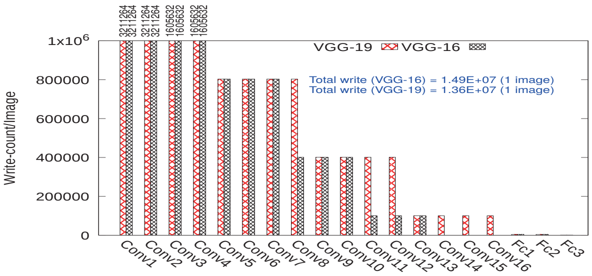


Fig. 7: Write reduction on PCM as all the intermediate values are written in DRAM.

3) Area analysis: Since memory industry is extremely area-sensitive, the area overhead needs to be considered when integrating additional logic near the memory. After implementing the Hydra module in Verilog, we use the Innovus from Cadence tool-set to obtain the post-synthesis area estimates. Each Hydra unit got placed on a 0.32 *mm* $\times$ 0.32 *mm* square box at 15 *nm* technology. This leads to an area overhead of 1.6 mm^2 in total for all 16 Hydra units. Compared to the 80 mm^2 area of a DRAM [6], this area overhead is only around 2%.

C. Comparison with previous accelerators

We compare our Hydra architecture with two popular DRAM-based processing-in-memory frameworks, DRISA [14], and DrAcc [4]. These architectures aim to modify the sub-array of commodity DRAM to exploit bit operation capability. Special rows are introduced to perform shift, NOT, and XOR operations in DrACC. In contrast, DRISA proposes to reorganize the bank and sub-array dimensions of DRAM for in-memory operations. Both DRISA and DrACC adapt the concept of concurrently activating multiple rows to realize the in-DRAM operations. Two limitations in this concept are: (1) the process variation in the sense amplifiers, and (2) high energy consumption stemming from the multi-row activations. The process variations often lead to logical operation failure. A 25% process variation can lead to around 26% failure in the logical operations [20]. Unlike DrAcc, DRISA addresses the problem by restructuring the sub-array that is different from the highly optimized commodity DRAM sub-array design. However, we use the sub-array only for traditional load-store operations, leading to no sub-array-level modification for the proposed work. The entire computations are done in a separate module, Hydra, integrated near the chip I/O. Multiplication operations (MUL) are crucial as they occupy the major portion of inference computation. It is also feasible to adopt quantized models to eliminate MULs in mobile/IoT devices. However, there is a non-negligible classification error of around 7% for ImageNet top-1 accuracy, between binary neural network [2] and the best case [25]. It is challenging to support the important multiplication operations in the PIM design. DrAcc skips the multiplications using ternary weight neural networks, while DRISA uses 1-bit weights for the inference. Instead of skipping multiplications, we perform them in MAC units,

TABLE II: Comparison with prior DRAM-based accelerators

	DrAcc [4]	DRISA [14]	Hydra
Perf. (FPS/ms)	0.3/3282 (VGG-16) 0.25/3933 (VGG-19)	0.3/3283 (VGG-16) 0.25/3933 (VGG-19)	6.61/151.32 (VGG-16) 5.21/192.03 (VGG-19)
Power (W)	2	98	3.63
Area (mm^2)	0.01	65.2	2.7
Efficiency (Fr./watt)	0.15	0.0031	1.82

making this suitable for the requirement of high accuracy. As shown in Table II, we obtain around 20x performance gain compared to both the state-of-the-art works because of the additional MAC-based logic used in Hydra. Additionally, the proposed system also leverages two-level of parallelism: (1) inter-chip and (2) intra-chip. The parallelisms and the additional MAC units provide high performance for the proposed Hydra-based system. Table II summarizes the quantitative key design specifications of all the accelerated architectures. The reported results for the state-of-the-art works are obtained from DrAcc [4]. On the comparable CMOS technology node, Hydra is more power and area efficient compare to DRISA, as shown in Table II. The area/power overhead of Hydra is higher than DrAcc because of the additional logic for processing. However, the efficiency (Frame/watt) of Hydra is substantially higher than both DRISA [14] and DrAcc [4], as shown in Table II.

V. Conclusion

We propose Hydra, a near hybrid memory accelerator for inference. We store the models in the denser PCM banks and prefetch them into the local buffers in parallel to the inference tasks. We place one Hydra module per DRAM chip to leverage inter-chip parallelism. We also exploit intra-chip parallelism by performing CONV/pool operations at 32 neuron positions of a channel in parallel to others. The placement of Hydra eliminates the intermediate writes of inference on the PCM, leading to enhanced lifetime. We also outperform the state-of-the-art works in terms of performance. Additionally, our system does not restrict the execution of non-quantized networks, unlike the previous works. The overall benefits of our proposed system make the Hydra design lucrative for practical implementation.

References

[1] E. Azarkhish, D. Rossi, I. Loi, and L. Benini, "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.

[2] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1," *arXiv preprint arXiv:1602.02830*, 2016.

[3] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: a large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.

[4] Q. Deng, L. Jiang, Y. Zhang, M. Zhang, and J. Yang, "DrAcc: a DRAM based accelerator for accurate cnn inference," in *Proceedings of the 55th Annual Design Automation Conference*, 2018, pp. 1–6.

[5] G. Dhiman, R. Ayoub, and T. Rosing, "PDRAM: a hybrid PRAM and DRAM main memory system," in *2009 46th ACM/IEEE Design Automation Conference*. IEEE, 2009, pp. 664–669.

[6] A. Farmahini-Farahani, J. H. Ahn, K. Morrow, and N. S. Kim, "NDA: Near-DRAM acceleration architecture leveraging commodity DRAM devices and standard memory modules," in *2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 2015, pp. 283–295.

[7] A. P. Ferreira, M. Zhou, S. Bock, B. Childers, R. Melhem, and D. Mossé, "Increasing PCM main memory lifetime," in *2010 Design, Automation & Test in Europe Conference & Exhibition (DATE 2010)*. IEEE, 2010, pp. 914–919.

[8] M. Gao, J. Pu, X. Yang, M. Horowitz, and C. Kozyrakis, "TETRIS: scalable and efficient neural network acceleration with 3D memory," in *Proceedings of the Twenty-Second International Conference on Architectural Support for Programming Languages and Operating Systems*, 2017, pp. 751–764.

[9] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[10] H. A. Khouzani, C. Yang, and J. Hu, "Improving performance and lifetime of DRAM-PCM hybrid main memory through a proactive page allocation strategy," in *The 20th Asia and South Pacific Design Automation Conference*. IEEE, 2015, pp. 508–513.

[11] D. Kim, J. Kung, S. Chai, S. Yalamanchili, and S. Mukhopadhyay, "NeuroCube: a programmable digital neuromorphic architecture with high-density 3D memory," *ACM SIGARCH Computer Architecture News*, vol. 44, no. 3, pp. 380–392, 2016.

[12] B. C. Lee, E. Ipek, O. Mutlu, and D. Burger, "Architecting phase change memory as a scalable DRAM alternative," in *Proceedings of the 36th annual international symposium on Computer architecture*, 2009, pp. 2–13.

[13] J. Lee, J. Chung, J. H. Ahn, and K. Choi, "Excavating the hidden parallelism inside DRAM architectures with buffered compares," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, no. 6, pp. 1793–1806, 2017.

[14] S. Li, D. Niu, K. T. Malladi, H. Zheng, B. Brennan, and Y. Xie, "DRISA: A DRAM-based reconfigurable in-situ accelerator," in *2017 50th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. IEEE, 2017, pp. 288–301.

[15] N. Niu, F. Fu, B. Yang, J. Yuan, F. Lai, and J. Wang, "WIRD: an efficiency migration scheme in hybrid DRAM and PCM main memory for image processing applications," *IEEE Access*, vol. 7, pp. 35 941–35 951, 2019.

[16] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," 2017.

[17] M. K. Qureshi, M. M. Franceschini, A. Jagmohan, and L. A. Lastras, "PreSET: Improving performance of phase change memories by exploiting asymmetry in write times," *ACM SIGARCH Computer Architecture News*, vol. 40, no. 3, pp. 380–391, 2012.

[18] M. K. Qureshi, V. Srinivasan, and J. A. Rivers, "Scalable high performance main memory system using phase-change memory technology," in *Proceedings of the 36th annual international symposium on Computer architecture*, 2009, pp. 24–33.

[19] S. Raoux, G. W. Burr, M. J. Breitwisch, C. T. Rettner, Y.-C. Chen, R. M. Shelby, M. Salinga, D. Krebs, S.-H. Chen, H.-L. Lung *et al.*, "Phase-change random access memory: A scalable technology," *IBM Journal of Research and Development*, vol. 52, no. 4.5, pp. 465–479, 2008.

[20] V. Seshadri, D. Lee, T. Mullins, H. Hassan, A. Boroumand, J. Kim, M. A. Kozuch, O. Mutlu, P. B. Gibbons, and T. C. Mowry, "Ambit: In-memory accelerator for bulk bitwise operations using commodity DRAM technology," in *2017 50th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. IEEE, 2017, pp. 273–287.

[21] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[22] S. Song, A. Das, O. Mutlu, and N. Kandasamy, "Enabling and exploiting partition-level parallelism (palp) in phase change memories," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 18, no. 5s, pp. 1–25, 2019.

[23] X. Wang, H. Liu, X. Liao, J. Chen, H. Jin, Y. Zhang, L. Zheng, B. He, and S. Jiang, "Supporting superpages and lightweight page migration in hybrid memory systems," *ACM Transactions on Architecture and Code Optimization (TACO)*, vol. 16, no. 2, pp. 1–26, 2019.

[24] W. Wei, D. Jiang, S. A. McKee, J. Xiong, and M. Chen, "Exploiting program semantics to place data in hybrid memory," in *2015 International Conference on Parallel Architecture and Compilation (PACT)*. IEEE, 2015, pp. 163–173.

[25] A. Zhou, A. Yao, Y. Guo, L. Xu, and Y. Chen, "Incremental network quantization: Towards lossless cnns with low-precision weights," *arXiv preprint arXiv:1702.03044*, 2017.