# Analysis of Conventional, Near-Memory, and In-Memory DNN Accelerators

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

Chandan Kumar Jha
*DFKI*, Germany
chandan.jha@dfki.de

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

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

***Abstract*—Various DNN accelerators based on Conventional compute Hardware Accelerator (CHA), Near-Data-Processing (NDP) and Processing-in-Memory (PIM) paradigms have been proposed to meet the challenges of inferencing Deep Neural Networks (DNNs). To the best of our knowledge, this work aims to perform the first quantitative as well as qualitative comparison among the state-of-the-art accelerators from each digital DNN accelerator paradigm. Our study provides insights into selecting the best architecture for a given DNN workload. We have used workloads of the MLPerf Inference benchmark. We observe that for Fully Connected Layer (FCL) DNNs, PIM-based accelerator is 21× and 3× faster than CHA and NDP-based accelerator respectively. However, NDP is 9× and 2.5× more energy efficient than CHA and PIM for FCL. For Convolutional Neural Network (CNN) workloads, CHA is 10% and 5× faster than NDP and PIM-based accelerator respectively. Further, CHA is 1.5× and 6× more energy efficient than NDP and PIM-based accelerators respectively.**

## I. Introduction

Deep Neural Networks (DNN) have evolved to do image classification, object detection, language processing, and recommendation tasks in mobile, edge, and data center applications [1]. These applications have specific latency and energy constraints based on the *scenarios* where these applications are deployed [2]. Since traditional computers with von Neumann architecture are not suitable for meeting these constraints due to the memory wall [3] and the power wall [4], various DNN accelerator architecture paradigms that perform exact digital computes have been proposed [5]–[10]. They can be broadly classified into Conventional Hardware Accelerators (CHA) [8], [9], Near Data Processors (NDP) [7], [9], [11], and Processing in Memory (PIM) Accelerators [6], [10] based on [12]. We conduct an in-depth qualitative and quantitative analysis of the three DNN accelerators - CHA, PIM, and NDP- for various DNN workloads to determine which accelerator is suitable for a given workload. The DNN workloads used here have two categoric layers - fully connected layer and convolutional layer. We compare different design metrics such as delay, energy, etc. For CNN workloads, CHA (Simba) outperforms NDP (Tetris) and PIM (AiM) in speed and energy. For FCL workloads like BERT and DLRM, PIM (AiM) is faster than CHA and NDP, but NDP is more energy efficient than both.

## II. State-of-the-art accelerators

To determine the best architecture within each paradigm, we use the following standard: (i) highest peak performance (TOPS), (ii) post-layout or hardware realized architectures, and (iii) comparable output quality results to the traditional hardware. Based on these criteria, we selected Simba [8], Tetris [7], and AiM [13] for CHA, NDP, and PIM, respectively.

*Simba* Simba consists of 16 Processing Elements (PEs) that each have 64 vectorized Multiply-Accumulate (MAC) units, resulting in a total of 1024 MACs for the whole architecture. Simba operates at 2 GHz and fetches data from external main memory for computation. Fig. 1 shows how the PEs are arranged in Simba Each PE has buffers that store the input activations, filters, and outputs of DNNs. These buffers help to reuse the data in DNNs, where different filters convolve with the same input word, and the same weight word applies to different input words. This way, the chip can cache the data needed for processing and reduce the number of accesses to external memory. Simba's external memory is a single DDR4 memory with 25GBps bandwidth in our model. However, accessing external memory consumes a lot of energy.

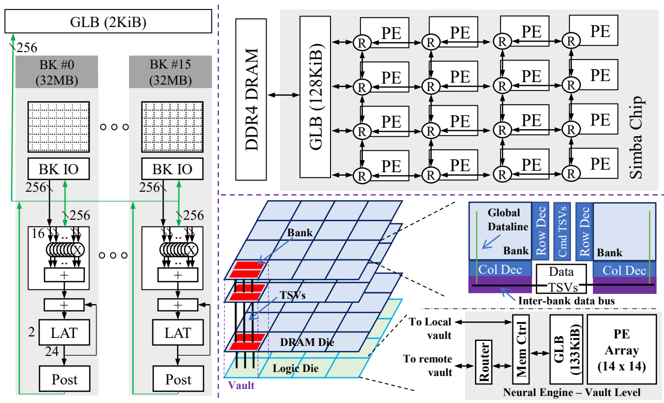

Fig. 1. Left: AiM Architecture [13]; Top Right: Simba Architecture [8] (CHA); Bottom Right: Tetris Architecture [7] (NDP)

*AiM* is based on GDDR6 DRAM and uses PIM paradigm. It has two dies with 16 banks of memory each. Each bank has a vector MAC with 16 multipliers that work at 1 GHz and use BFloat16. The access energy is low because the data is fetched from the bank and computed in the periphery. The input activation is moved to the Global Buffer (GLB) and shared by 16 Vector MACs. Each vector MAC sums up the products into a single word. The single words from 16 banks are combined into a new row of data and written back to any bank. This spatial arrangement is similar to Simba's PE organization and minimizes reads and writes. However, the MAC units in AiM are made with a DRAM process and are less energy efficient and slower than CMOS chips.

*Tetris* Tetris combines the benefits of Simba and AiM by stacking DRAM over a logic chip and connecting them with Through-Silicon-Vias (TSVs). The TSVs offer high bandwidth and low energy for data transfer. The logic chip has area and TDP constraints due to stacking. Tetris has 16 vaults with 3136 MAC units in total that work at 500 MHz to limit power consumption.

## III. Experimental Setup and Results

We extend Timeloop [2] to model the three SOTA architectures with hardware values [7], [8], [13]. We use 45nm model and CACTI for buffers. We optimize mapping for latency and energy for each architecture and workload. We get metrics layer by layer for the DNNs. We abbreviate CNN layers and FCLs of CNN as AN (AlexNet), LN (LeNet), MN (MobileNet), RN (ResNet), VN (VGG16); fully connected networks as BF (BERT), DF (DLRM), LF (LSTM); architectures as A (AiM), S (Simba), T (Tetris). We calculate weighted average of all layers for results.

### A. Latency

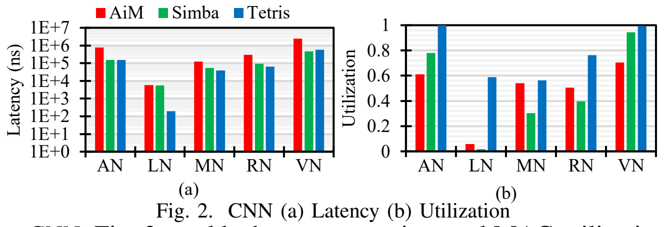

Fig. 2. CNN (a) Latency (b) Utilization

*CNN:* Fig. 2a and b show average time and MAC utilization for CNN layers on various architectures. Simba is 5× and 10% faster than AiM and Tetris for CNNs. The latency gap is due to three factors: (i) Simba and Tetris can do 4 and 3 MACs in the time AiM does 1 MAC. (ii) Simba has less bandwidth but more data reuse with large on-chip buffer. (iii) Simba uses 50% of its high-frequency MACs for CNN workloads as in Fig. 2b. *Simba is better than Tetris and AiM for latency-sensitive CNN applications.*

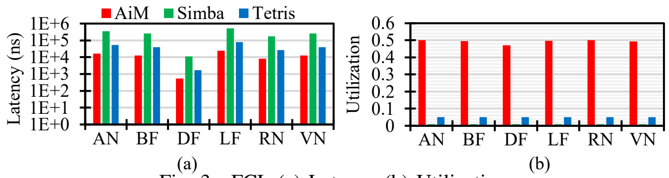

Fig. 3. FCL (a) Latency (b) Utilization

*FCL:* Fig. 3a and b show average time and MAC utilization for FCL layers on various architectures. AiM is 21× and 3× faster than Simba and Tetris for FCL. The latency gap is due to two factors: (i) FCL has low data reuse potential, i.e., one compute/data word - causing LLM bandwidth bottleneck in Simba. Simba has 1% utilization as in Fig 3b. (ii) Tetris has 5% utilization as each vault has low bandwidth (2GBps per MAC in AiM vs. 0.08GBps per MAC in Tetris). AiM has 50% utilization as GLB-bank transfer is not possible during MAC operation. *AiM is better than Simba and Tetris for latency-sensitive FCL applications.*

### B. Energy

*CNN:* Fig. 4 and Fig. 5 show average energy and energy split per computation for CNN on the left. Simba is the most energy-efficient for CNNs. Simba uses 1.4× and 6.2× less energy than Tetris and AiM for CNNs. The energy gap is due to: (i) Simba reduces buffer reads and writes per computation by sharing input data, accumulating eight products to a single word, and having a larger accumulation buffer. This lowers buffer access energy. (ii) Simba has 5× fewer off-chip memory accesses with efficient shared buffering, but each access is 10× more costly. Still, these accesses account for 64% of total energy per compute in Simba compared to 23% and 30% in AiM and Tetris. (iii) In AiM, BF16 computation makes MAC operation energy 4× higher than Simba, and buffer access energy 10× higher due to constant reading from row buffer and swapping from small accumulation latch. In AiM, LLM access energy is 3.5× more than Tetris due to direct reading of operands from row buffer at each band while Tetris uses SRAM registers.

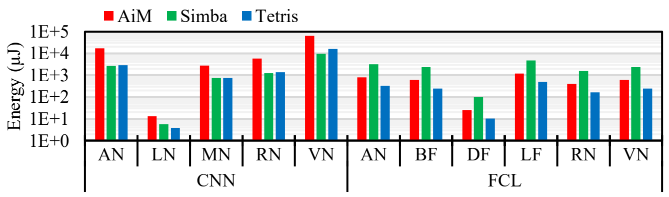

Fig. 4. Average energy for processing each layer of CNN and FCL

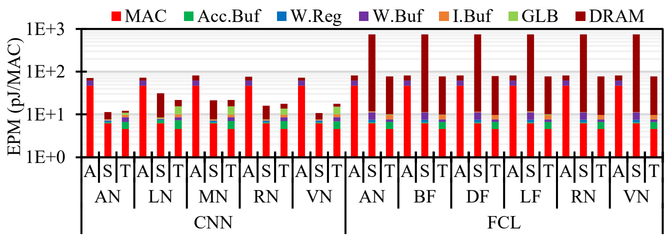

Fig. 5. Log scale: Split up of Energy per Operation for CNN and FCL

*FCL:* Fig. 4 and Fig. 5 show average energy per layer and per compute for FCL. Tetris is the most energy-efficient for FCL. Tetris uses 2.4× and 9.6× less energy than AiM and Simba. Simba uses 10× more energy to fetch data from main memory than AiM for FCL. This and higher MAC energy make Simba use 9.6× more energy than Tetris. In AiM, LLM access energy is 3.5× less than Tetris. But MAC unit energy is 10× higher than Tetris due to BFloat multiplication, which dominates AiM's energy.*Simba and Tetris are better for energy-sensitive CNNs and FCLs applications, respectively.*

## IV. Conclusion

We perform an in-depth analysis of state-of-the-art DNN hardware accelerators from Conventional Hardware Accelerator (CHA), Near-Data Processing (NDP), and Processing-in-Memory (PIM) architecture paradigms to identify the fastest and the most energy-efficient architecture paradigm for different ML workloads from MLPerf benchmark suite. We identified that CHA (Simba), on average, has 10% and 5× speedup than NDP (Tetris) and PIM (AiM) for CNN workloads. In comparison, PIM (AiM) has, on average 21× and 3× speedup than CHA and PIM for workloads with Fully Connected Layers (FCL) such as BERT and DLRM. This inversion is due to the high data reuse potential in CNN workloads, negligible data reuse potential in FCL, and the limited bandwidth of the last-level memory in CHA. NDP is the most efficient paradigm and consumes 2.5× and 9.6× lower energy than PIM and CHA for FCL. For CNN workloads, CHA is 1.4× and 6.2× more energy efficient than NDP and PIM.

## Acknowledgment

This work is supported through grants received from Science and Engineering Research Board (SERB), Government of India, under SERB-CRG grant CRG/2018/005013, SERB-MATRICS grant MTR/2019/001605, and SERB-SUPRA grant SPR/2020/000450, and funds received for YFRF Visvesvaraya PhD fellowship from MEITY and Semiconductor Research Corporation (SRC) through contracts 2020-IR-3005 and 2020-IR-2980, and is partially supported through Ashoka University startup and Huawei Technologies India grants.

## References

[1] P. Mattson, C. Cheng, C. Coleman, G. Diamos, P. Micikevicius, D. Patterson, H. Tang, G.-Y. Wei, P. Bailis, V. Bittorf *et al.*, "Mlperf training benchmark," *arXiv preprint arXiv:1910.01500*, 2019.

[2] A. Parashar, P. Raina, Y. S. Shao, Y.-H. Chen, V. A. Ying, A. Mukkara, R. Venkatesan, B. Khailany, S. W. Keckler, and J. Emer, "Timeloop: A systematic approach to dnn accelerator evaluation," in *2019 IEEE international symposium on performance analysis of systems and software (ISPASS)*. IEEE, 2019, pp. 304–315.

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

[4] X. Guo, E. Ipek, and T. Soyata, "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] Y. Chen, Y. Xie, L. Song, F. Chen, and T. Tang, "A survey of accelerator architectures for deep neural networks," *Engineering*, vol. 6, no. 3, pp. 264–274, 2020.

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

[7] 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.

[8] B. Zimmer, R. Venkatesan, Y. S. Shao, J. Clemons, M. Fojtik, N. Jiang, B. Keller, A. Klinefelter, N. Pinckney, P. Raina *et al.*, "A 0.32–128 tops, scalable multi-chip-module-based deep neural network inference accelerator with ground-referenced signaling in 16 nm," *IEEE Journal of Solid-State Circuits*, vol. 55, no. 4, pp. 920–932, 2020.

[9] G. F. Oliveira, J. Gómez-Luna, L. Orosa, S. Ghose, N. Vijaykumar, I. Fernandez, M. Sadrosadati, and O. Mutlu, "Damov: A new methodology and benchmark suite for evaluating data movement bottlenecks," *arXiv preprint arXiv:2105.03725*, 2021.

[10] J. Gómez-Luna, I. E. Hajj, I. Fernandez, C. Giannoula, G. F. Oliveira, and O. Mutlu, "Benchmarking a new paradigm: An experimental analysis of a real processing-in-memory architecture," *arXiv preprint arXiv:2105.03814*, 2021.

[11] P. Das, A. Joshi, and H. K. Kapoor, "Hydra: A near hybrid memory accelerator for cnn inference," in *2022 Design, Automation and Test in Europe Conference & Exhibition (DATE)*, 2022, pp. 1017–1022.

[12] M. He, C. Song, I. Kim, C. Jeong, S. Kim, I. Park, M. Thottethodi, and T. Vijaykumar, "Newton: A dram-maker's accelerator-in-memory (aim) architecture for machine learning," in *2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. IEEE, 2020, pp. 372–385.

[13] S. Lee, K. Kim, S. Oh, J. Park, G. Hong, D. Ka, K. Hwang, J. Park, K. Kang, J. Kim, J. Jeon, N. Kim, Y. Kwon, K. Vladimir, W. Shin, J. Won, M. Lee, H. Joo, H. Choi, J. Lee, D. Ko, Y. Jun, K. Cho, I. Kim, C. Song, C. Jeong, D. Kwon, J. Jang, I. Park, J. Chun, and J. Cho, "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 *2022 IEEE International Solid- State Circuits Conference (ISSCC)*, vol. 65, 2022, pp. 1–3.