

Memory-Oriented Design-Space Exploration of Edge-AI Hardware for XR Applications

Vivek Parmar¹, Syed Shakib Sarwar², Ziyun Li², Hsien-Hsin S. Lee², Barbara De Salvo^{2†}, and Manan Suri^{1†}

¹Indian Institute of Technology Delhi, ²Meta Reality Labs Research

[†]Corresponding Authors: barbarads@meta.com; manansuri@ee.iitd.ac.in

ABSTRACT

Low-Power Edge-AI capabilities are essential for on-device extended reality (XR) applications to support the vision of Metaverse. In this work, we investigate two representative XR workloads: (i) Hand detection and (ii) Eye segmentation, for hardware design space exploration. For both applications, we train deep neural networks and analyze the impact of quantization and hardware specific bottlenecks. Through simulations, we evaluate a CPU and two systolic inference accelerator implementations. Next, we compare these hardware solutions with advanced technology nodes. The impact of integrating state-of-the-art emerging non-volatile memory technology (STT/SOT/VGSOT MRAM) into the XR-AI inference pipeline is evaluated. We found that significant energy benefits ($\geq 24\%$) can be achieved for hand detection (IPS=10) and eye segmentation (IPS=0.1) by introducing non-volatile memory in the memory hierarchy for designs at 7nm node while meeting minimum IPS (inference per second). Moreover, we can realize substantial reduction in area ($\geq 30\%$) owing to the small form factor of MRAM compared to traditional SRAM.

KEYWORDS

Extended Reality, Deep Neural Networks, Edge Computing, Non-Volatile Memories

ACM Reference Format:

Vivek Parmar¹, Syed Shakib Sarwar², Ziyun Li², Hsien-Hsin S. Lee², Barbara De Salvo^{2†}, and Manan Suri^{1†}. 2023. Memory-Oriented Design-Space Exploration of Edge-AI Hardware for XR Applications. In *Proceedings of tinyML Research Symposium (tinyML Research Symposium '23)*. ACM, New York, NY, USA, 7 pages.

1 INTRODUCTION

Extended reality (XR), *i.e.*, virtual, augmented, and mixed reality is fast emerging as a key technology paradigm for the future edge and mobile systems in the incoming era of Metaverse or Omniverse. XR technology has a wide variety of applications in entertainment, communication, advertising, education, healthcare, defense, robotics, smart manufacturing, human-machine interaction, etc. XR applications are becoming more computationally intensive [7] which poses new challenges for designing portable XR devices and systems.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

tinyML Research Symposium '23, March 2023, Burlingame, CA

© 2023 Copyright held by the owner/author(s).

The current generation portable XR devices depend extensively on high-performance compute servers to perform the heavy-lifting computation due to limitation on local device's power, compute capability, and memory capacity. This approach, however, has disadvantages such as (i) patchy and non-seamless user experiences, (ii) data transfer/network overheads, and (iii) user privacy and security concerns. Further, the explosive growth and success of techniques such as deep learning for computer vision have made computationally intensive AI-based techniques a natural use case for future XR systems [7]. The projected specifications of some current and future generation XR devices are shown in Table 1 [7]. In certain vision-based use cases, very high-resolution (~200 MP) and high frame rates (>90 Hz) are required at modest power budgets (<1W). In this study, we perform detailed architectural design-space exploration and DTCO (design technology co-optimization) for building optimized portable XR systems while tackling some of these concerns. Our key contributions and the novel aspects are: (i) Two XR-specific computer vision AI workloads were analyzed: (a) Hand detection using DetNet with FPHAB¹ dataset and (b) Eye segmentation using UNet with OpenEDS dataset. Both models were evaluated based on full precision and post-training quantization. (ii) Benchmarking of the XR-AI applications was performed on three architectures including a general-purpose Intel-based CPU architecture and two systolic accelerator architectures: NVidia's Simba, and MIT's Eyeriss. (iii) Technology scalability study at process nodes of 28nm, 22nm, and 7nm for all three architectures was conducted and their respective EDP (energy delay product) trends were investigated. (iv) Non-volatility was introduced into the XR compute pipeline by replacing SRAM with emerging MRAM devices (STT/SOT/VGSOT MRAM) for all three architectures through two variants: (a) P0 (Weight Buffer and Global Weight Buffer replaced by MRAM), (b) P1 (all memory replaced by MRAM). (v) Compared to SRAM-only architecture, memory power savings of 27% with area savings of ~16% were observed for P0 variants. Correspondingly for P1 variants, memory power savings of 24% and area savings of ~34% were observed compared to SRAM-only variants.¹

2 ANALYSIS ON REPRESENTATIVE XR-AI WORKLOADS

In this section, we present algorithmic approaches for training networks used in XR-AI inference workloads of interest followed by details regarding quantization-based inference optimization. Hand detection [5, 6] and eye segmentation [4] have been heavily used as part of VR and AR headset deployment. Segmentation of ocular

¹IIT Delhi obtained and used the FPHAB dataset.

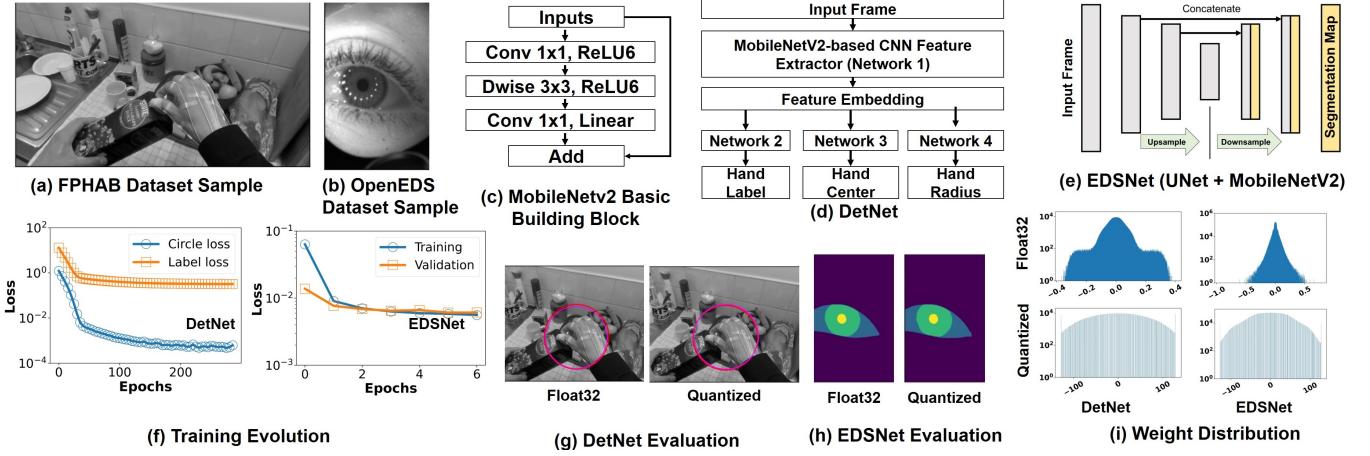


Figure 1: Sample images from datasets: (a) FPHAB and (b) OpenEDS. (c) MobileNetV2 building block (Inverted Residual Bottleneck [13]). (d) DetNet. (e) EDSNet (UNet model + MobileNetV2 Backbone). (f) Training loss evolution. (g) DetNet evaluation, on sample image, with FP32 and INT8 precision. Red circle shows ground truth and purple shows predicted. (h) EDSNet evaluation, on sample image, with FP32 and INT8 precision. (i) Trained and quantized weight distributions for both networks.

Table 1: Projected specs of state-of-the-art XR devices [7].

Metric	HTC Vive Pro	Ideal VR	Microsoft HoloLens2	Ideal AR
Resolution (MP)	4.6	200	4.4	200
Refresh rate (Hz)	90	90-144	120	90-144
Motion-to-photon latency (ms)	<20	<20	<9	<5
Power (W)	N/A	1-2	>7	0.1-0.2

biometric traits of the eye region such as pupil, iris, sclera have been extensively used for studying eye movements as well as performing gaze estimation [4], making it imperative to XR display designs when optimizing users’ experiences. Similarly, vision-based hand tracking has been explored and adopted by commercial platforms such as Meta’s Oculus Quest, HTC’s VIVE, and Microsoft’s Hololens as a convenient and low-friction input for XR devices to enable a seamless and less strenuous user experience. Hence, both applications are considered representative by this work for benchmarking on a variety of compute platforms for XR-AI.

2.1 Dataset Description

FPHAB. Since XR devices employ ego-centric/first-person view, we use the First Person Hand Action Benchmark (FPHAB) dataset from [5] for training the hand detection network. The dataset consists 100K frames of 45 daily hand action categories, involving 26 different objects in several hand configurations. Training and validation sets consist of 52,868 and 52,212 frames, respectively. The dataset provides annotation in the form of 3D locations of 21 key joints of a hand estimated using a motion capture glove. When mapped to a 2D frame, they result in 21 keypoints.

OpenEDS. For eye segmentation applications, we use the OpenEDS 2019 dataset in [4]. OpenEDS was collected from voluntary participants of ages between 19 and 65. The dataset contains a total of 12,759 eye images with corresponding annotation masks for key eye

regions such as (a) Eyelid, (b) Iris, and (c) Pupil. The dataset is then divided into three splits, namely, (i) training (8,916), (ii) validation (2,403), and (iii) testing (1,440). The sample images from FPHAB and OpenEDS datasets are illustrated in Fig. 1(a) and Fig. 1(b).

2.2 Network Training and Quantization

All our neural network training experiments were performed using PyTorch. Optimized neural network architectures such as MobileNet [13] have been adopted for XR applications, e.g., detecting hand gestures [6]. A key building block in such architectures known as inverted residual bottleneck (IRB) is shown in Fig. 1(c). The IRB helps reduce the memory footprint during inferences by not fully materializing large intermediate tensors (using depth-wise separable convolution, i.e., two layers in place of a single convolution layer), thus reducing the frequency of main memory accesses. To perform hand detection, we trained the DetNet which is composed of a MobileNetV2-based feature extractor and three regression networks to estimate the center, the radius, and the labels of the tracked hand. The network shown in Fig. 1(d) performs a bounding circle detection to enable the tracking of the joint movement. To train the DetNet, we first converted the keypoint annotations of FPHAB dataset to bounding circles. The center of each circle was estimated by computing the mean of x and y coordinates for each keypoint, while the radius was estimated as the maximum distance in XY plane between the center and all keypoints. The DetNet was trained over 300 epochs using AdamW optimizer. We used a combination of two loss components for the overall network training: (i) Circle loss, i.e., the loss in MSE (mean square error) for predicting center and radii of bounding circles for both hands and (ii) Label loss, i.e., the cross-entropy loss for predicting left hand or right hand. The training progress for each loss component is shown in Fig. 1(f). The Circle loss is calculated as the weighted sum of the center and the radius MSE losses with a higher weight given to the center. As depicted in Fig. 1(f), the Circle loss achieves MSE values around 10^{-3} within 200 epochs. To perform eye segmentation, we trained

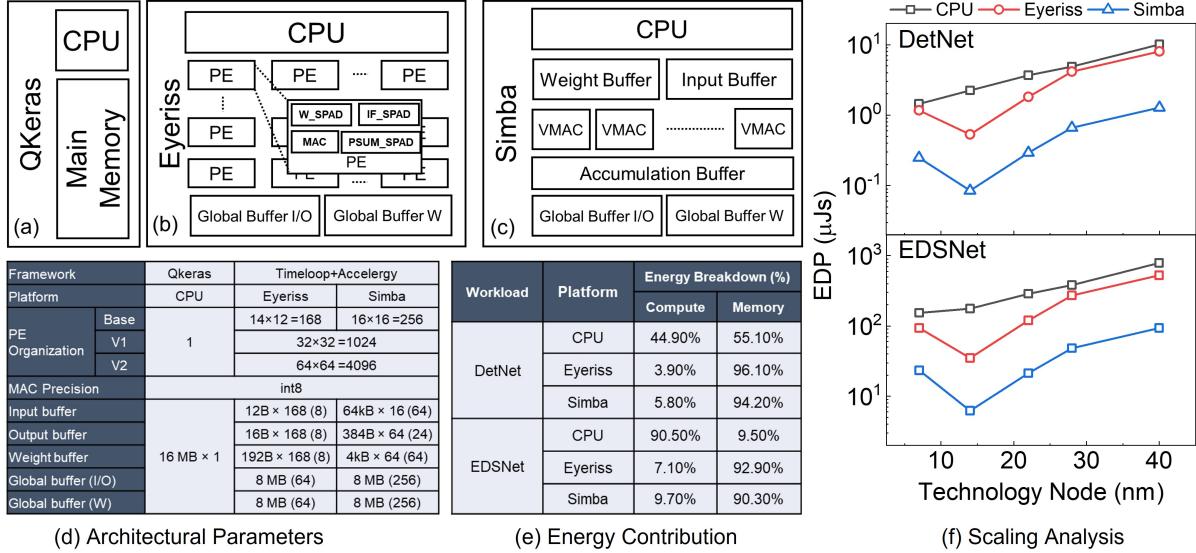


Figure 2: Simulated: (a) CPU + Memory in QKeras, (b) Eyeriss PE [1] and (c) Simba [16] in Timeloop. (d) Specification of simulated architectures used in the study. Numbers mentioned in bracket indicate bus size. (e) Energy Breakdown of simulated architectures. (f) Estimated EDP for inference of DetNet and EDSNet. SRAM-only variant was estimated at 45 nm for CPU and 40 nm for Eyeriss/Simba. Scaling estimates on other nodes are based on [8, 14].

EDSNet–UNet model [12] with MobileNetV2 backbone (Fig. 1(e))—using the "segmentation models" library [20]. The training was performed using Adam optimizer with DiceLoss over six epochs. The training progress is shown in Fig. 1(f). The loss value converges within three epochs indicating high efficiency of the trained feature extractor. Since most optimized edge AI hardware platforms can benefit from using lower precision (e.g., INT8), we performed post-training quantization on both models using NVIDIA’s TensorRT library. The evaluation of full precision and quantized models of DetNet on samples from the dataset are visualized in Fig. 1(g). Similarly, the segmentation results on a sample image using both FP32 and quantized INT8 models of EDSNet are shown in Fig. 1(h). The weight histograms for trained and quantized models for both networks are shown in Fig. 1(i). The quantized model shows a more smooth and uniform weight distribution with discrete levels. This further helps model compression by opening possibilities for weight sharing across layers [1]. The satisfactory inference results for both networks, with INT8 quantization, is exploited for hardware exploration discussed in the following sections.

3 IMPLEMENTATION ON EDGE-AI ACCELERATORS

We benchmark our XR-AI workloads on three simulated architectures illustrated in Fig. 2: (i) a generic CPU [2] and two systolic inference accelerators: (ii) Eyeriss [1], and (iii) Simba [16]. These architectural simulations help us to investigate the roles of various important design parameters such as datapath, operation mapping, parallelism, and memory hierarchy as described in Fig. 2(d). The key difference between Eyeriss and Simba is in their memory organization. While Eyeriss heavily relies on localized memory for every PE (processing element), Simba utilizes shared buffers across rows in the form of input buffer, weight buffer, and accumulation buffer.

For architectural workload mapping and network simulations, we used the following three frameworks: QKeras [2], Timeloop [10], and Accelergy [19]. In the case of QKeras (CPU), models were first translated to Keras followed by quantization using QKeras library with energy estimation based on the operation mapping to a CPU instruction set. QKeras maps the workload to a pure CPU architecture and provides energy estimates at 45nm node [2]. QKeras also allows choices of memory configurations, they are (a) SRAM-only (b) SRAM+DRAM with writeback (c) DRAM-only. For the current study, we use SRAM-only configuration for the memory.

Timeloop was used to estimate the cycle-wise operation mapping of the two neural network workloads on the systolic PEs based on Eyeriss (row-stationary) and Simba (weight-stationary). For using Timeloop, we exported the models from torch using the `pytorch2timeloop` converter. We performed the following modifications on baselines Simba and Eyeriss to make them more relevant for the XR-AI use cases. First, DRAM was completely removed from both accelerators and SRAM global buffer size was chosen as per workload requirement shown in Fig. 2(d). While both SRAM and DRAM are volatile memory technologies, DRAM offers a lower area/cost in contrast to that SRAM offers latency and energy benefits which are critical for such applications. Secondly, we employed Aladdin’s 40nm standard cell library as a reference in place of the original 45nm one provided by Accelergy. The adoption of 40nm cell library enabled INT8 support for Eyeriss in place of the default INT16 MAC operations. Moreover, since the 40nm library offers multiple versions of modules in adders/multipliers/register, it enables DTCO through Accelergy on the basis of energy-latency trade-offs. CACTI [15] is used to estimate the energy for various

SRAM buffers shown in Fig. 2(b) and (c). The estimated EDP for inference of both workloads—hand detection and eye segmentation—is shown in Fig. 2(f). Apart from the baseline DRAM-free variants at 45nm/40nm, we also projected energy scaling for more advanced nodes (28nm, 22nm, and 7nm) for all three architectures. Energy and latency scaling factors used for the analysis were derived from [8, 14]. Scaling from the baseline technology node (45nm for CPU, 40nm for Simba/Eyeriss) leads to an energy reduction of up to 4× across all architectures. While the systolic accelerators may have significant benefits in terms of latency, it can be observed that energy costs increase significantly as compared to a baseline CPU. In case of 7nm, Simba and Eyeriss show similar energy dissipations for EDSNet while in case of DetNet Simba shows energy savings of 11% compared to Eyeriss. The discrepancy at 7nm observed for EDSNet can be attributed to the memory-intensive nature of the workload which benefits from row-stationary architecture of Eyeriss.

4 PROPOSED NVM-BASED ENHANCEMENT

In previous sections, we explored the implication of network architecture and computing platforms in terms of EDP. In addition to the absolute energy depicted in Fig. 2(f), Fig. 2(e) further analyzes the energy dissipation for the systolic architectures (Eyeriss and Simba) and indicates that memory power dissipation is far more significant than that of compute, leaving more room for optimization. One such optimization already included was the removal of DRAM. Furthermore, from literature it is evident that some XR-AI workloads are highly asymmetric in terms of their temporal compute requirements; *i.e.*, AI compute may not be executed at every cycle or uniformly with time, but rather in a sporadic manner [6]. Such peculiar compute requirements can benefit from active power-gating (*e.g.*, normally-off computing) of the edge-AI accelerators to extend the battery life. An essential component required to implement power-gated/normally-off edge systems is non-volatile memory (NVM). NVM enables quick wake-up from off/sleep modes without the need of energy-hungry and time-consuming data reloads to SRAM or main memory [17]. A major benefit of these NVMs is observed in silicon area due to use of additional BEOL process or 3D integration. As shown in [18], cell area reductions of up to 1.3x, 2.3x, and 2.5x can be achieved for SOT-, VGSOT-, and STT-MRAM over their high-density SRAM counterpart. Moreover, recent progress of emerging magneto-resistive/spintronic NVM (STT-MRAM, SOT-MRAM, etc.) has led to device performance comparable to that of SRAM [18]. To assess this prospect, we performed a detailed analysis of energy dissipation of the aforementioned architectures for the two XR-AI workloads after including two state-of-the-art NVM devices, STT and SOT, in the XR-AI compute pipelines.

The temporal operation cycle of the simulated XR-AI pipeline is shown in Fig. 3(a). It involves the execution modes in following sequence: (i) Accelerator wakeup (WU), (ii) Frame Acquisition or frame load (FA), (iii) AI Inference, and (iv) Power-Gating of Accelerator. The memory type (SRAM or NVM) used in the system will have a direct impact on the overall latency and energy. A pipeline that uses only volatile SRAM will follow the operation cycle shown in Fig. 3(b)-(i), while an alternate pipeline that uses NVM in Fig. 3(b)-(ii) can be powered-off during the intervals after

performing inference without the need of any rewrite. Option to go in power-off mode due to non-volatility of memory leads to energy savings. We propose two strategies, P0 and P1 mappings shown in Fig. 3(c), to adopt NVM-based pipelines in the edge devices for the XR-AI workloads. The per-inference cycle memory operation breakdown for AI inference is shown in Fig. 3(c). In the proposed P0 mapping as shown in Fig. 3(c)-(ii), we introduce NVM (STT and SOT) only for the weight memory. In a more aggressive variant P1 mapping, we replace all SRAM memory buffers with NVM as illustrated in Fig. 3(c)-(iii).

5 RESULTS AND DISCUSSION

To estimate the energy for the proposed variants P0 and P1, MRAM and SRAM macro energy characterization from recent literature is used (7nm [18], 28nm [17]) along with our compute/MAC energy analysis. The total workload energy was estimated by using operation counts based on Timeloop+Accelergy and QKeras simulations. A 64-bit memory bit-width is assumed for CPU while Timeloop employs memory bit-widths specific to the architecture (see Fig. 2(d)). Fig. 3(d) presents a comprehensive analysis of energy trends for both XR-AI workloads on nine different simulated architectural variants (three flavors each for CPU, Eyeriss and Simba) for two technology nodes (28nm and 7nm). For each of the three architectures, three memory flavors are considered, *i.e.*, SRAM only, P0: SRAM+MRAM, and P1:MRAM only. NVM technology used for 7nm estimates is VGSOT-MRAM [18] in place of STT-MRAM. Since the parameters used for VGSOT-MRAM are based on highly-scaled device estimates, a scaling factor based method was employed to first energy scaling in terms of SRAM. Subsequently SRAM to VGSOT-MRAM, scaling factor is employed based on literature data[18]. Some key observations from single inference energy analysis (shown in Fig. 3(d)) are listed below.

- Both P0 and P1 variants show higher energy dissipation compared to SRAM-only case at 7nm for the systolic accelerators, whereas for CPU the energy dissipation is nearly equivalent irrespective of workload.
- P1 variants show higher energy dissipation for all architectures and workloads across both nodes. This can be attributed to the asymmetric energies for read and write operation shown by MRAM as compared to SRAM.
- At 28nm, P0 variants of all architectures show energy savings compared to SRAM-only case for both workloads while a reverse trend exists at 7nm. This can be attributed to the difference in read energy costs demonstrated by STT-MRAM and VGSOT-MRAM *i.e.* VGSOT-MRAM is optimized for write while STT-MRAM is optimized for read.

The detailed energy breakdown in terms of compute and memory operations (read/write) is shown in Fig. 4. For all workloads and architectures based on P0 configuration and P1 at 7nm, the memory read energy dominates the memory write energy. In case of P1-28nm, this trend reverses for all architectures and workloads except for Simba with EDSNet workload. This can be attributed to the weight-stationary dataflow of Simba which results in reduced memory fetches for weights. Compute energy dominates over memory for CPU and the trend is reversed for both systolic accelerators.

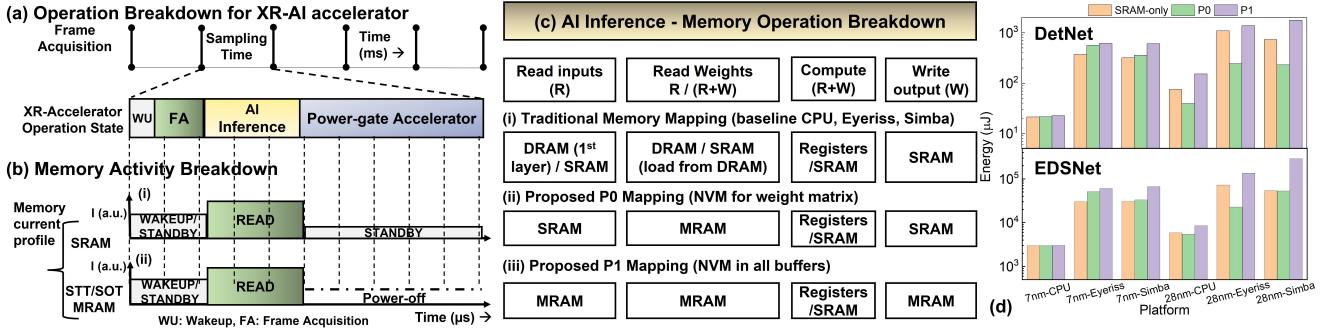


Figure 3: (a) Operation breakdown for XR-AI accelerator (b) Memory activity profile during XR-AI workload execution: (i) SRAM (ii) NVM. (c) Breakdown of memory specific operations in the AI Inference mode. Proposed NVM introduction strategies: (ii) P0 (SRAM+MRAM) and (iii) P1 (MRAM-only). (d) Single inference energy dissipation for 9 simulated architectural variants on DetNet and EDSNet.

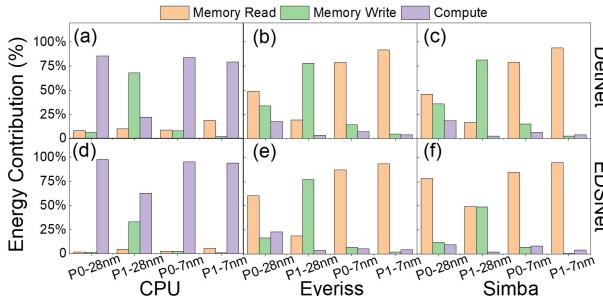


Figure 4: Simulated energy breakdown in terms of memory and compute for NVM-based architectural variants for DetNet on: (a) CPU (b) Eyeriss (c) Simba, and EDSNet on: (d) CPU (e) Eyeriss (f) Simba.

This can be attributed to the sequential computation dataflow employed by the CPU thus reducing unnecessary memory fetches. For P1-7nm, the memory read energy becomes overwhelmingly dominant ($\approx 50\times$) in comparison to memory write energy for all architectures and workloads. This can be attributed to the fact that the VGSOT-device used for 7nm is more optimized for write as opposed to read.

Next, we analyze the benefits in terms of the area by introducing NVM for the systolic accelerator architectures at 7nm node. To perform area estimation, compute area was scaled as per scaling factor derived from Deepscale [14]. For memory area estimates of SRAM, we utilized CACTI config files used by Accelergy with Fin-CACTI [15] tool. Next, area scaling factors based on the feature size of a single bit-cell were derived for SRAM and VGSOT-MRAM [18]. Using internal CACTI computations for multiple sizes of SRAM memory, periphery area factors were derived to estimate overheads at subarray, MAT, and Bank level, respectively [15]. Using the above mentioned methodology, area estimates were derived for both P0 and P1 variants as summarized in Table 2. While P0 variants show marginal benefits in area ($\approx 16\%$), P1 variants show 34% area savings as compared to the standard SRAM-only architecture. A key reason for smaller area benefits of P0 variants can be attributed to the periphery area overhead for small memory macros. This was

Table 2: Estimation of Area Benefits on Systolic Accelerators using Proposed P0 and P1 variants at 7nm node.

Architecture	7 nm Area (mm^2)			Area savings	
	SRAM-only	P0	P1	P0	P1
Simba	2.89	2.41	1.88	16.56%	34.97%
Eyeriss	2.56	2.11	1.67	17.52%	34.98%

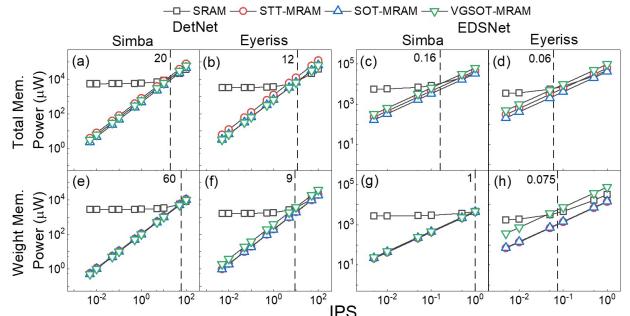


Figure 5: Simulated memory power vs. IPS benchmarking for proposed architectural variants utilizing SRAM, STT, SOT, and VGSOT devices. (a,b,e,f) correspond to Simba while (c,d,g,h) correspond to Eyeriss. For each case, IPS cross-over points w.r.t SRAM and VGSOT are indicated in the plots. Any IPS value below cross-over point would lead to energy-savings while using NVM. Plots (a-d) (top row) are for P1 variants while (e-h) (bottom row) are for P0 variants. For P0 variants, cross-over points are limited based on maximum frequency supported by the memory architecture.

especially true based on the current workloads where weight memory could be optimized leading to requirements of 12 kB for storage of model weights. However for more complex workloads involving video streams, weight memory may emerge as a significant factor leading to better savings for P0.

To analyze the impact of the asymmetric temporal compute profile of the workloads, we estimate memory power (total, weight, I/O buffer) as a function of hypothetical inference event frequency / IPS (inference per second). This metric is a direct function of the required frame rate of the application and can also assist in modeling workload for an accelerator receiving input streams from multiple

Table 3: IPS Analysis summary for proposed architectures using PE configuration v2 (64×64).

XR-AI Workload	Architecture	Inference Latency (ms)		P_{Mem} Savings @ IPS_{min}	
		P0	P1	P0	P1
DetNet $IPS_{min}=10$	Simba	0.34	0.42	27%	31%
	Eyeriss	0.86	0.86	-4%	9%
EDSNet $IPS_{min}=0.1$	Simba	48.57	60.72	29%	24%
	Eyeriss	45.22	45.22	-15%	-26%

sensors since the focus is on the throughput of the accelerator. In this analysis, it is assumed that accelerators can be put to sleep (power-gated) during the intervals between the completion of an inference and the arrival of the next inference request. The standby current of memory is assumed to be 100× lower compared to the read current[11] with a wakeup time of 100μs. The memory power vs IPS estimates for 7nm node using SRAM and three spintronic devices (STT, SOT, and VGSOT) are shown in Fig. 5. Fig. 5(a-d) and Fig. 5(e-h) show the results of memory energy savings for P1 and P0 variants, respectively. The key results on memory power savings for all combinations are summarized in Table 3. In addition, the inference latency results shown in Table 3 reflect that Simba offers the best opportunity to exploit sleep time intervals. The latency numbers were based on the estimated cycle counts extracted from Timeloop [10] multiplied by the frequency of operation for accelerator. The base frequency of compute is derived from the physically realized chips of the accelerators [1, 16] scaled down to 7nm using DeepScale [14]. Operational frequency is primarily limited by memory. Hence, using peak workload-specific memory bandwidth requirements derived from Timeloop+Accelergy simulations, a relaxed operation frequency was estimated. Here we assume support for multi-cycle read and write operations using corresponding memory technology.

An important point to note here is that at 7nm, all memory technologies under consideration have very low read and write latencies (≤ 5 ns) equivalent to SRAM's [18], thus resulting in operations running at similar inference latencies as the SRAM-only case. Here, we fix application-specific inference throughput values (IPS_{min}) of ~ 10 and ~ 0.1 for hand detection and eye segmentation applications respectively, which in the extreme case may go up to ~ 40 and ~ 6 respectively [3, 9]. The key observations from Memory Power vs. IPS analysis are listed below:

- The noticeable differences in memory power for different spintronic devices (Fig. 5(a-d)) can be attributed to the differences in the read and write energy for each device type (STT, SOT, VGSOT), where VGSOT has the lowest write energy but higher read energy.
- In the case of P0 variants shown in (Fig. 5(e-h)), it can be observed that achievable cut-off IPS (IPS for which SRAM and MRAM variants show equal power dissipation) with VGSOT improves for Simba whereas it decreases for Eyeriss. This can be attributed to the smaller local weight buffers used by Eyeriss requiring increased read operations in the global weight-memory.
- P0 variants show a clear distinction in MRAM variants for the EDSNet workload which can be attributed to the increased

requirement of read operations in the weight memory due to the nature of the workload.

- While P0 variants of Simba outperform P1 variants in terms of achievable cut-off IPS (see Fig. 5b and Fig. 5f) this comes at the cost of increased power (see Table 3) and area (see Table 2). Furthermore, a hybrid memory architecture would lead to higher design complexity.

From the above analysis, it can be summarized that for the scaled nodes (7nm) P1 variant outperforms P0 and SRAM-only variants for DetNet workload in terms of memory power savings as well as area when operating at lower inference rates. However, this trend is reversed in case of a read-intensive workload such as EDSNet that heavily uses the input buffer and thus reduces savings from VGSOT-MRAM which is more write-optimized. P1 variants also incur the cost of slightly higher inference latency ($\approx 20\%$). However, this can be considered inconsequential with regards to the application since the latency of the P1 variants can very well satisfy the minimum IPS requirement of the application for real-world use cases. Using the accelerators with uniquely different dataflows, we can observe that while row-stationary may be beneficial for energy savings in a conventional CMOS architecture, weight-stationary dataflow leads reduced stress on memory bandwidth. This in turn facilitates the applicability of NVM in the memory hierarchy. This makes a case for switching to higher proportion of on-chip NVM with aggressive device scaling. However, based on the nature of workload and IPS requirement of the application, a complete replacement of on-board volatile memory with NVM may not be the optimal choice as NVM write latency might limit the computation speed. Furthermore, given the asymmetric energy dissipation trends of read and write operations for state-of-the-art NVM devices the power benefits maybe limit. Hence, based on the exact nature of the workload (*i.e.* memory read-dominated or memory write-dominated), one needs to carefully fine-tune the proportion of the splits between NVM and SRAM to achieve the optimal results.

6 CONCLUSION

We present a detailed study on two XR-AI workloads (hand detection and eye segmentation). We first present results for network training and quantization. To perform more extensive design exploration, simulations were performed for CPU and systolic accelerators using QKeras and Timeloop+Accelergy frameworks with node-scaling analysis. Finally, we propose memory-oriented DTCO based on the use of different types of the emerging MRAM devices. We also analyze the energy benefits of introducing non-volatility in the XR compute pipeline with respect to the inference activity rates at 7nm node. When MRAM NVM was introduced in the memory hierarchy, memory energy savings $\geq 24\%$ were observed for hand detection (at $IPS = 10$) and eye segmentation (at $IPS=0.1$), respectively. Additionally, MRAM replacing SRAM leads to substantial area reduction ($\geq 30\%$) due to the high density feature of MRAM technology.

REFERENCES

- [1] Yu-Hsin Chen, Tushar Krishna, Joel S. Emer, and Vivienne Sze. 2017. Eyeriss: An Energy-Efficient Reconfigurable Accelerator for Deep Convolutional Neural Networks. *IEEE JSSC* 52, 1 (2017), 127–138.

- [2] Claudiorn N. Coelho, Aki Kuusela, Shan Li, Hao Zhuang, Jennifer Ngadiuba, Thea Klaeboe Arrestad, Vladimir Loncar, Maurizio Pierini, Adrian Alan Pol, and Sioni Summers. 2021. Automatic heterogeneous quantization of deep neural networks for low-latency inference on the edge for particle detectors. *Nature Machine Intelligence* (jun 2021), 1–12.
- [3] Yu Feng, Nathan Goulding-Hotta, Asif Khan, Hans Reyserhove, and Yuhao Zhu. 2022. Real-Time Gaze Tracking with Event-Driven Eye Segmentation. *CoRR* abs/2201.07367 (2022). arXiv:2201.07367
- [4] Stephan J. Garbin, Yiru Shen, Immo Schuetz, Robert Cavin, Gregory Hughes, and Sachin S. Talathi. 2019. OpenEDS: Open Eye Dataset. *CoRR* abs/1905.03702 (2019). arXiv:1905.03702
- [5] Guillermo Garcia-Hernando, Shixin Yuan, Seungryul Baek, and Tae-Kyun Kim. 2018. First-Person Hand Action Benchmark With RGB-D Videos and 3D Hand Pose Annotations. In *CVPR*. 409–419.
- [6] Shangchen Han, Beibei Liu, Randi Cabezas, Christopher D. Twigg, Peizhao Zhang, Jeff Petkau, Tsz-Ho Yu, Chun-Jung Tai, Muzaffer Akbay, Zheng Wang, Asaf Nitzan, Gang Dong, Yuting Ye, Lingling Tao, Chengde Wan, and Robert Wang. 2020. MEgATrack: monochrome egocentric articulated hand-tracking for virtual reality. *ACM Trans. Graph.* 39, 4 (2020), 87.
- [7] Muhammad Huzaifa, Rishi Desai, Xutao Jiang, Joseph Ravichandran, Finn Sinclair, and Sarita V. Adve. 2020. Exploring Extended Reality with ILLIXR: A New Playground for Architecture Research. *CoRR* abs/2004.04643 (2020). arXiv:2004.04643
- [8] Norman P. Jouppi, Doe Hyun Yoon, Matthew Ashcraft, Mark Gottsch, Thomas B. Jablin, George Kurian, James Laudon, Sheng Li, Peter C. Ma, Xiaoyu Ma, Thomas Norrie, Nishant Patil, Sushma Prasad, Cliff Young, Zongwei Zhou, and David A. Patterson. 2021. Ten Lessons From Three Generations Shaped Google’s TPUs: Industrial Product. In *ISCA-48*. 1–14.
- [9] Adarsh Kowdle, Christoph Rhemann, Sean Ryan Fanello, Andrea Tagliasacchi, Jonathan Taylor, Philip L. Davidson, Mingsong Dou, Kaiwen Guo, Cem Keskin, Sameh Khamis, David Kim, Danhang Tang, Vladimir Tankovich, Julien P. C. Valentin, and Shahram Izadi. 2018. The need 4 speed in real-time dense visual tracking. *ACM Trans. Graph.* 37, 6 (2018), 220:1–220:14. <https://doi.org/10.1145/3272127.3275062>
- [10] Angshuman Parashar, Priyanka Raina, Yakun Sophia Shao, Yu-Hsin Chen, Victor A. Ying, Anurag Mukkara, Rangharajan Venkatesan, Brucek Khailany, Stephen W. Keckler, and Joel S. Emer. 2019. Timeloop: A Systematic Approach to DNN Accelerator Evaluation. In *ISPASS*. 304–315.
- [11] R. Ranica, N. Planes, O. Weber, O. Thomas, S. Haendler, D. Noblet, D. Croain, C. Gardin, and F. Arnaud. 2013. FDSOI process/design full solutions for ultra low leakage, high speed and low voltage SRAMs. In *2013 Symposium on VLSI Technology*. IEEE, T210–T211.
- [12] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-Net: Convolutional Networks for Biomedical Image Segmentation. In *LNCS*. Springer International Publishing, 234–241.
- [13] Mark Sandler, Andrew G. Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. 2018. MobileNetV2: Inverted Residuals and Linear Bottlenecks. In *CVPR*.
- [14] Satyabrata Sarangi and Bevan Baas. 2021. DeepScaleTool: A Tool for the Accurate Estimation of Technology Scaling in the Deep-Submicron Era. In *ISCAS*. 1–5.
- [15] Alireza Shafaei, Yanzhi Wang, Xue Lin, and Massoud Pedram. 2014. FinCACTI: Architectural Analysis and Modeling of Caches with Deeply-Scaled FinFET Devices. In *ISVLSI*. IEEE Computer Society, 290–295. <https://doi.org/10.1109/ISVLSI.2014.94>
- [16] Yakun Sophia Shao, Jason Cemons, Rangharajan Venkatesan, Brian Zimmer, Matthew Fojtik, Nan Jiang, Ben Keller, Alicia Klinefelter, Nathaniel Ross Pinckney, Priyanka Raina, Stephen G. Tell, Yanqing Zhang, William J. Dally, Joel S. Emer, C. Thomas Gray, Brucek Khailany, and Stephen W. Keckler. 2021. Simba: scaling deep-learning inference with chiplet-based architecture. *Commun. ACM* 64, 6 (2021), 107–116.
- [17] Manan Suri, Abhishek Gupta, Vivek Parmar, and Kang Ho Lee. 2019. Performance Enhancement of Edge-AI-Inference Using Commodity MRAM: IoT Case Study. In *IMW*. 1–4.
- [18] Y.C. Wu, K. Garello, W. Kim, M. Gupta, M. Perumkunil, V. Kateel, S. Couet, R. Carpenter, S. Rao, S. Van Beek, K.K. Vudy Sethu, F. Yasin, D. Crotti, and G.S. Kar. 2021. Voltage-Gate-Assisted Spin-Orbit-Torque Magnetic Random-Access Memory for High-Density and Low-Power Embedded Applications. *Phys. Rev. Applied* 15 (Jun 2021), 064015. Issue 6.
- [19] Yannan Nellie Wu, Joel S. Emer, and Vivienne Sze. 2019. Accelergy: An Architecture-Level Energy Estimation Methodology for Accelerator Designs. In *ICCAD*. 1–8.
- [20] Pavel Yakubovskiy. 2019. Segmentation Models. https://github.com/qubvel/segmentation_models.