

# QUILL: An Algorithm–Architecture Co-Design for Cache-Local Deformable Attention

Hyunwoo Oh\*, Hanning Chen\*, Sanggeon Yun\*, Yang Ni<sup>†</sup>, Wenjun Huang\*, Tamoghno Das\*, Suyeon Jang\*, and Mohsen Imani\*

\*University of California, Irvine, <sup>†</sup>Purdue University Northwest  
Email: {hyunwoo, m.imani}@uci.edu

**Abstract**—Deformable Transformers achieve state-of-the-art object detection, but deformable attention maps poorly to hardware due to irregular memory access and low arithmetic intensity. We present QUILL, a schedule-aware accelerator that makes MSDeformAttn cache-local and single-pass. QUILL’s *Distance-based Out-of-Order Querying (DOOQ)* reorders queries by spatial proximity, enabling a look-ahead, double-buffered prefetch that overlaps memory and compute. QUILL also uses a fused MSDeformAttn pipeline that performs interpolation, Softmax, aggregation, and output projection in one pass, avoiding intermediate spills and keeping small tensors on-chip. Implemented in RTL and evaluated end-to-end, QUILL achieves up to 7.29× higher throughput and 47.3× better energy efficiency than an RTX 4090, and improves throughput/energy efficiency over prior accelerators by 3.26–9.82× / 2.01–6.07×. With mixed precision, accuracy stays within  $\leq 0.9$  AP of FP32 across Deformable and Sparse DETR variants. By converting sparsity into locality and locality into utilization, QUILL delivers consistent end-to-end gains.

**Index Terms**—Deformable Attention, Vision Transformer, Hardware/Software Co-design, DETR, ML Accelerator.

## I. INTRODUCTION

Detection transformers (DETR) [1] have gotten widespread attention for their end-to-end object detection capabilities, often outperforming CNN-based detectors in simplicity and adaptability [2]. As a result, DETR-based models have seen rapid adoption in tasks such as recognition and multimodal AIs [1]–[7]. However, original DETR variants suffer from high compute cost and slow convergence, mainly due to dense attention mechanisms that obscure gradient flow from input features [8].

Deformable DETR [8] addresses these issues with multi-scale deformable attention (MSDeformAttn), sampling only a handful of spatial locations per query. Inspired by deformable convolutions [9], this design significantly reduces FLOPs and accelerates training. Consequently, as shown in Fig. 1, MSDeformAttn underpins state-of-the-art detector modules [10]–[16] and has been adopted in emerging tasks such as video restoration [17] and large vision–language models [18].

Despite MSDeformAttn’s algorithmic efficiency, practical deployments on GPUs expose notable latency and throughput drawbacks [19]–[21], limiting adoption in real systems [22]. Our profiling on an RTX 4090 highlights four observations:

- 1. Fewer FLOPs, higher latency (Fig. 2).** Although the backbone accounts for most FLOPs, the *deformable transformer dominates run time*. MSDeformAttn’s mathematically sparse pattern *fragments memory access*, triggering frequent stalls.
- 2. Pruning reduces FLOPs but not latency proportionally (Fig. 3(a)).** Query pruning [23] lowers FLOPs roughly with the keeping ratio  $\rho$  [14], but latency gains remain modest (e.g., even at  $\rho=0.1$ ) due to scattered sampling overheads.

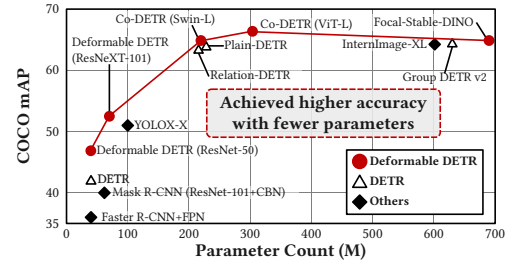


Fig. 1: Landscape of SOTA object detection models.

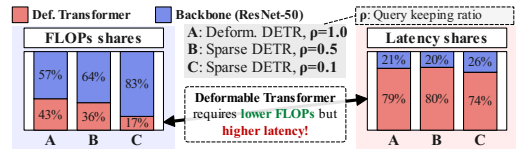


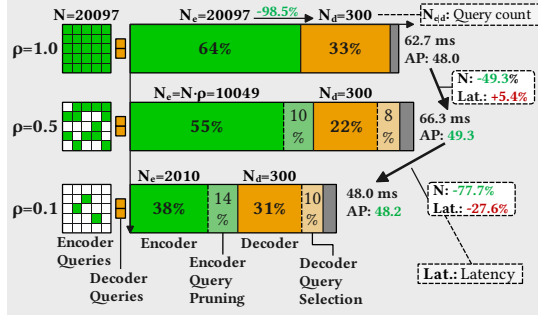
Fig. 2: FLOPs vs. latency on RTX 4090 comparing the backbone to the deformable transformer block.

**3. Decoder cost remains substantial even with 98.5% fewer queries (Fig. 3(a), top).** The decoder uses a tiny set of queries (e.g., 300; a **98.5% reduction** vs. the encoder’s 20,097), yet it still **accounts for  $\sim 33\%$  of runtime**, indicating an **order-of-magnitude higher per-query cost**. Deformable DETR variants exacerbate this by increasing  $N_d$  (e.g.,  $N_d=900$  [11], [16]).

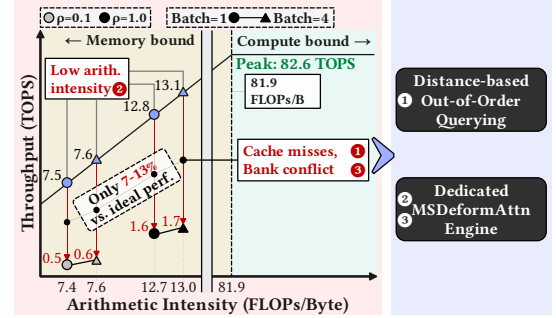
**4. Three fundamental bottlenecks (Fig. 3(b)).** A roofline view identifies cache misses (❶), low arithmetic intensity (❷), frequent bank conflicts (❸) [20] as dominant causes of **low PE utilization** on current platforms.

These results reveal a mismatch between MSDeformAttn’s sparse behavior and modern GPU architectures, motivating an **algorithm–architecture co-design** that preserves model outputs while reorganizing execution for locality and utilization.

Most transformer accelerators target standard attention where accesses are denser and more regular [24]–[28]. In contrast, *MSDeformAttn is inherently irregular*, sampling from scattered references across multi-level features. Recent MSDeformAttn accelerators [20], [21] leave critical gaps: (i) **DEFA** [20] processes queries in a fixed sequential order and relies on a sliding-window cache, which is **ill-suited to the decoder and pruned encoder** where sparsity dominates, leading to **severe cache misses**; moreover, it runs bilinear interpolation, Softmax, and linear layers as **separate micro-kernel passes**, forcing full memory load–store between passes. (ii) **UEDA** [21] **struggles to scale to standard Deformable DETR settings** (e.g.,  $N_e=20,097$ ,  $D=256$ ) due to **FPGA resource constraints** and limited algorithmic leverage. To our knowledge, no prior art (1) **addresses sparse query irregularity at runtime** and (2) **unifies encoder, decoder, and FFN** in one execution path.



(a) GPU latency breakdowns on Deformable Transformer



(b) Roofline analysis on Def. Trans. and proposed solutions

Fig. 3: GPU performance breakdowns of Deformable DETR and query-pruned models on RTX 4090. (a) Latency across internal blocks. (b) Roofline view of bottlenecks and mitigation levers.

We present **QUILL**, an *algorithm–architecture co-design* that turns deformable attention into cache-friendly work while preserving model semantics. Unlike fixed traversal/sliding-window schemes [20] or FPGA-restricted pipelines [21], QUILL introduces a runtime, proximity-aware reordering that restructures execution for locality. Our contributions are:

- **DOOQ runtime (query-locality scheduling with aligned prefetch).** *Distance-based Out-of-Order Querying (DOOQ)* reorders queries by spatial proximity to raise cache hit rate. Its lookup window exposes predictable addresses, enabling *double-buffered* prefetch aligned to the schedule, overlapping fetch with compute and cutting cache-miss stalls (1).
- **Fused MSDeformAttn execution path.** A *single-pass* path that fuses sampling, weighting (Softmax), and aggregation to lift arithmetic intensity (2) and avoid intermediate stores; conflict-aware buffers curb bank conflicts (3), addressing inefficiencies in prior designs [20].
- **RTL prototype & full-stack evaluation.** A synthesizable RTL of the full accelerator (DOOQ scheduler, feature cache, fused core, gather–scatter, GEMM I/F), functionally verified against a reference MSDeformAttn, with cycle-accurate simulation, post-synthesis PPA, and end-to-end throughput/energy vs. RTX 4090 and DEFA (incl. ablations on  $w_d$ ,  $r$ ).

Across Deformable DETR variants, QUILL achieves up to **7.29× higher throughput** and **47.3× better energy efficiency** than an RTX 4090 GPU. Compared to DEFA [20], QUILL improves throughput by **3.26–9.82×**, attributable to DOOQ’s locality gains together with the fused execution path.

## II. PRELIMINARIES & HARDWARE CHALLENGES

We give a **hardware-centric view of MSDeformAttn** and explain why its *sparse, data-dependent* access pattern dominates latency on existing platforms. We then show how common sparsification (e.g., *query pruning*) further degrades locality, motivating our algorithm–architecture co-design.

### A. MSDeformAttn: A Hardware View

MSDeformAttn [8] computes, for query  $q$ , reference point  $p$  and spatial embedding feature map  $x$ ,

$$\text{MSDeformAttn}(q, p, x) = \sum_{m=1}^M W_m \cdot \left[ \sum_{l=1}^L \sum_{k=1}^K A_{qmlk} \cdot W'_m \cdot x(\hat{p}_q + \Delta p_{qmlk}) \right] \quad (1)$$

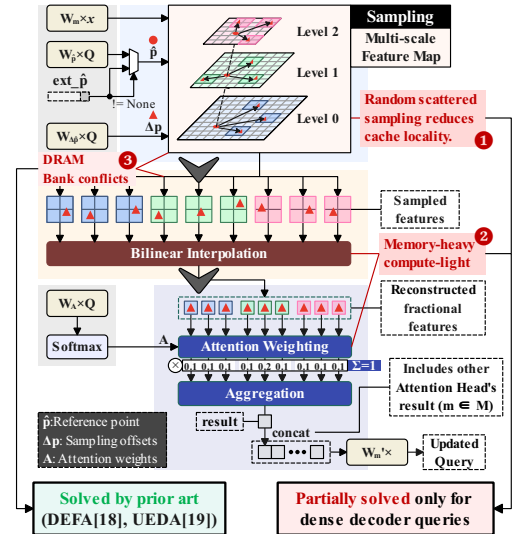


Fig. 4: MSDeformAttn dataflow linked to Eq. (1), with hardware bottlenecks 1–3 marked at the memory touchpoints: 1 cache misses from scattered samples; 2 low arithmetic intensity (few ops per byte); 3 bank conflicts around fractional  $2 \times 2$  fetches.

where  $m, l, k$  index head, feature level, and sampling point. Each query gathers a few *fractional* locations per head across multi-level features (via  $2 \times 2$  bilinear interpolation), applies attention weights  $A$ , and aggregates the result. FLOPs are modest, but the *irregular, multi-level gathers* fragment memory access (Fig. 4), yielding 1 cache misses, 2 low arithmetic intensity, and 3 frequent bank conflicts [20].

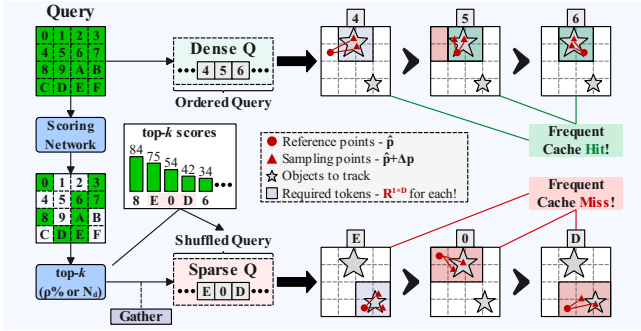
### B. Query Pruning Hurts Data Locality

As shown in Fig. 5, Sparse DETR [14] prunes encoder queries to the top- $\rho\%$  to reduce FLOPs, and decoder variants select the top- $N_d$  tokens from encoder outputs. These steps *scatter and shuffle* the surviving queries in space, further degrading spatial/temporal locality. As a result, latency on GPUs does not scale with the FLOP reduction [23], [29]–[31].

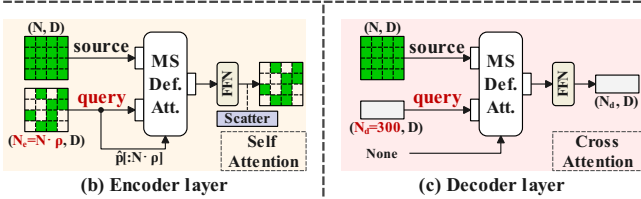
### C. Fundamental Bottlenecks and Implications

Across encoder and decoder (and under pruning), deformable transformers face **three** hardware bottlenecks:

- 1 **Cache misses:** Scattered, cross-level gathers defeat locality.
- 2 **Low arithmetic intensity:** Few arithmetic ops per fetched byte leave compute underutilized.
- 3 **Bank conflicts:**  $2 \times 2$  patch fetches for fractional token interpolation elevate collision risk [20].



(a) Query pruning concept and impact on data locality



(b) Encoder layer

(c) Decoder layer

Fig. 5: **Sparsity amplifies irregular access.** (a) Top- $k$  pruning cuts FLOPs but scatters queries, reducing cache hit rate. (b) Sparse encoder (top- $\rho\%$ ). (c) Decoder using top- $N_d$  tokens. Both induce less cache-friendly patterns.

These factors limit throughput and efficiency on existing platforms, even FLOPs are reduced. Closing this gap requires a *runtime* that **restores locality** and makes next accesses predictable enough to *drive double-buffered (ping-pong) prefetch*, paired with a fused execution path that avoids redundant load-store traffic—principles we adopt in our co-design.

### III. ALGORITHM-ARCHITECTURE CO-DESIGN

QUILL targets the three bottlenecks from Sec. II by turning deformable attention into cache-friendly work. At a high level, DOOQ chooses a spatially local execution order for queries; those predicted next addresses drive a region-based prefetch into an alternate cache buffer; the fused core consumes the current buffer and completes interpolation, weighting (Softmax), aggregation, and projection without spilling intermediates; and small, reusable tensors stay on-chip while a light gather-scatter makes sparse I/O contiguous. The surrounding dense layers run on standard systolic GEMMs. Unlike fixed traversal or sliding-window caches [20] and FPGA-restricted pipelines [21], the schedule adapts online and exposes predictable next addresses that a prefetcher can exploit. We refer to this DOOQ→look-ahead→ping-pong path as a schedule-aware prefetch loop.

Our co-design follows two principles. First, use a *runtime, content-agnostic schedule* (DOOQ) that restores locality without changing model semantics and exposes predictable next addresses for prefetch. Second, execute MSDeformAttn in a *single pass* so locality gains translate directly into higher arithmetic intensity and fewer stalls.

#### A. (a) Feature Caching Unit: DOOQ with aligned prefetch

The feature cache (Fig. 6, (a)) forms a closed loop: select the next queries, prefetch their regions, and compute on the current buffer. DOOQ supplies the order; a small selector realizes it; and double buffering turns predictable addresses into overlap between memory and compute. The loop is illustrated in Fig. 7, and summarized in Alg. 1.

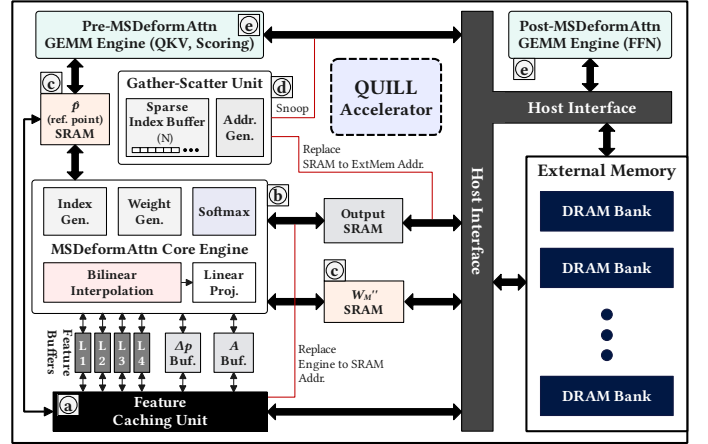


Fig. 6: **Overview of QUILL.** The design has two pillars: (a) a feature caching unit driven by DOOQ and aligned double-buffered prefetch to mitigate cache misses (1); and (b) a fused MSDeformAttn core that executes interpolation, Softmax, aggregation, and projection in one pass to lift arithmetic intensity (2) and curb conflict hotspots (3). Support blocks include (c) on-chip reuse SRAMs for small, frequently reused tensors, (d) a light gather-scatter path for sparse I/O, and (e) standard GEMM engines for surrounding dense layers. *Core of QUILL: DOOQ with aligned prefetch, and a fused single-pass core (vs. fixed/sliding traversal).*

Within a lookup window of size  $w_d$ , DOOQ prefers the next queries whose reference points  $\hat{p}$  are spatially closest to the current one, so adjacent queries reuse many of the same feature lines across levels. If  $M(q)$  is the set of cache lines needed by query  $q$ , miss-driven stall for execution order  $\sigma$  is

$$T_{\text{stall}}(\sigma) = \sum_{i=1}^{n-1} t_{\text{fetch}} \cdot |M(q_{\sigma(i+1)}) \setminus M(q_{\sigma(i)})|, \quad (2)$$

and DOOQ acts directly on this set difference. Algorithm 1 (line 2) performs the greedy nearest-neighbor selection in  $\ell_1$ ; Fig. 7 depicts the corresponding distance+sorting unit. Unlike fixed traversal or sliding-window caches [20], the order adapts online to the current distribution of  $\hat{p}$  while keeping hardware simple (we use  $\ell_1$  to avoid squaring and square roots with negligible impact on reuse).

Candidates pass through a cyclic bitonic sorter (Fig. 7, “distance+sorting”) that reuses compare-and-swap elements over multiple cycles [32]; the per-query slack (about  $D/p_d$  cycles) amortizes its cost for typical  $w_d$ . Area scales linearly with  $w_d$  (time-multiplexed  $O(w_d)$  comparators over  $O(\log^2 w_d)$  steps), and timing fits within the  $D/p_d$  slack. This guarantees a one-step look-ahead: when the core begins  $q_{\sigma(i)}$ , the cache already knows the reference points for  $q_{\sigma(i+1)}$ .

That look-ahead drives supply. The prefetch in Alg. 1 (line 3) issues a regional fetch around each chosen  $\hat{p}$  into the alternate buffer while the core consumes the current one (the ping-pong path in Fig. 7). At the handoff, only incremental lines are fetched if the next query stays within the prefetched region. Under ping-pong, the residual stall per step is  $\max\{0, t_{\text{fetch}}(q_{\sigma(i+1)} | w_d) - t_{\text{comp}}(q_{\sigma(i)})\}$ , and DOOQ shrinks the first term by maximizing the overlap between  $M(q_{\sigma(i)})$  and  $M(q_{\sigma(i+1)})$ . With average hit rate  $h$  and regional reuse  $\rho$ , the covered miss time per step is approximately  $\min\{t_{\text{comp}}, (1 - h\rho) t_{\text{fetch}}\}$ .

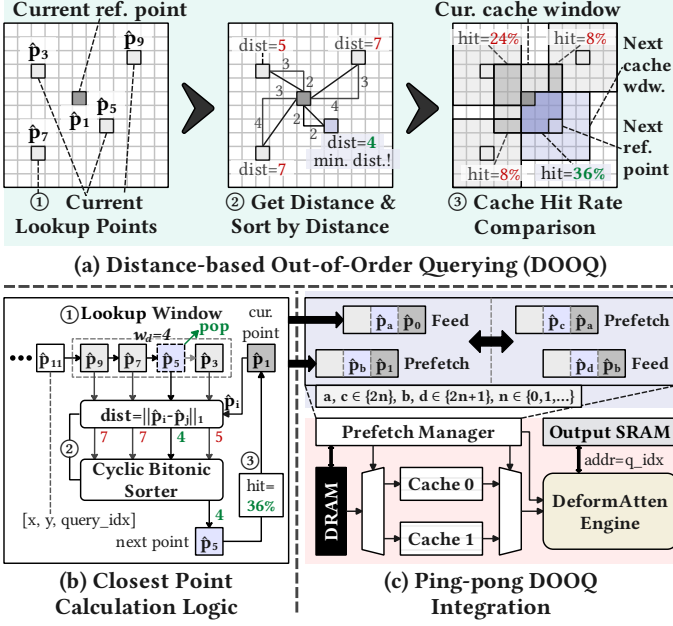


Fig. 7: **DOOQ and aligned prefetch.** DOOQ chooses spatially closest queries within a window; a lightweight distance+sorting (cyclic bitonic) unit selects the next set; and a ping-pong prefetch overlaps memory with compute using addresses from the DOOQ window. Together these form the schedule-aware prefetch loop: schedule  $\rightarrow$  look-ahead  $\rightarrow$  overlapped prefetch.

Flow control is elastic: the emit/update in Alg. 1 (line 4) advances the loop; FIFOs handle backpressure, a small per-level victim path recovers rare off-region offsets, and query IDs preserve semantic order.

#### Algorithm 1 DOOQ (schedule-aware prefetch loop)

**Require:** lookup window  $\mathcal{W}$  of reference points  $\{\hat{p}\}$ , current query  $q_i$ , region radius  $r$

- 1: **while**  $\mathcal{W} \neq \emptyset$  **do**
- 2:  $q^* \leftarrow \arg \min_{q \in \mathcal{W}} \|\hat{p}_q - \hat{p}_{q_i}\|_1$  selection
- 3: **PrefetchRegion** $(\hat{p}_{q^*}, r)$  to alternate buffer
- 4: **Emit** $(q^*)$ ;  $\mathcal{W} \leftarrow \mathcal{W} \setminus \{q^*\}$ ;  $q_i \leftarrow q^*$  update
- 5: **end while**

#### B. (Ⓟ) Fused MSDeformAttn Core

The core turns locality into throughput by avoiding intermediate spills; we execute MSDeformAttn in a single pass. Folding the projections into  $W_m'' = W_m \cdot W_m'$  yields

$$MSDeformAttn(q, p, x) = \sum_{m=1}^M W_m'' \cdot \left[ \sum_{l=1}^L \sum_{k=1}^K A_{qmlk} \cdot x(\hat{p}_q + \Delta p_{qmlk}) \right] \quad (3)$$

so interpolation, Softmax, aggregation, and projection proceed without intermediate writes that depress arithmetic intensity. Index and weight generators produce  $2 \times 2$  coordinates and bilinear weights  $w_{ij}$ ; the same  $w_{ij}$  broadcasts across the  $D$ -wide vector, which keeps the multiplier count low. The  $2 \times 2$  gather plus interpolation uses four multipliers and three adders per element, with the small generators shared across samples over multiple cycles. Softmax runs in-core to avoid traffic, using shared exp/add hardware (Padé-based exponential [33])

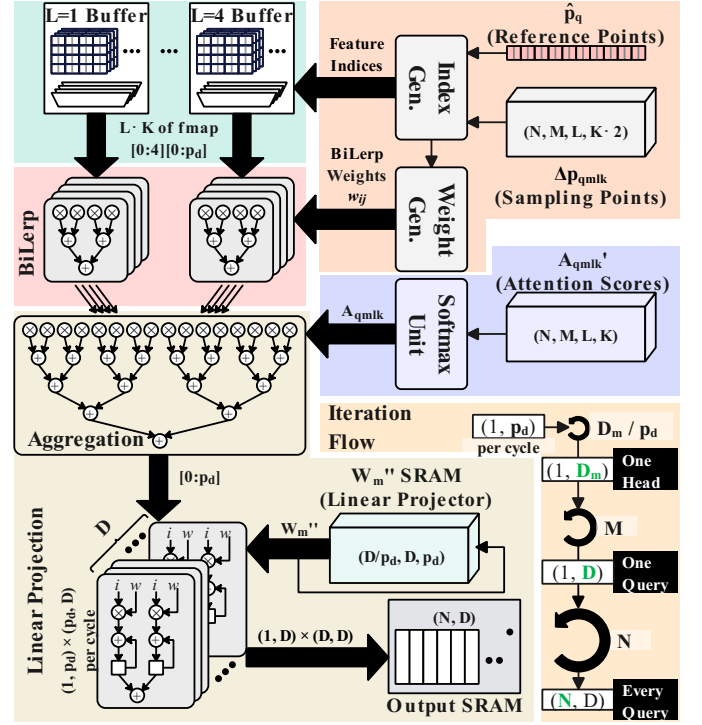


Fig. 8: **Fused MSDeformAttn core.** Interpolation, Softmax, aggregation, and projection ( $W_m''$ ) are executed in one pass; level buffers are organized to serve  $2 \times 2$  fetches with minimal collisions.

across cycles. Aggregation feeds a projector backed by on-chip storage for  $W_m''$ , which is loaded once per block and reused across queries. Level buffers are laid out so the common  $2 \times 2$  access pattern maps to distinct banks; the DOOQ schedule further reduces simultaneous contention across levels, trimming conflict hotspots without requiring large SRAMs. We reorder across queries only; within each query the  $L \times K$  sampling order is unchanged, and Softmax is computed per query, preserving semantics.

#### C. (Ⓞ–Ⓞ) Support blocks

Small, frequently reused tensors stay on-chip to cut redundant transfers and help arithmetic intensity: reference points  $\hat{p}$  (two coordinates per query) and the projector  $W_m''$  (sized by  $M$  and  $D$ ). If  $D$  or  $M$  grows,  $W_m''$  streams in stripes along  $D$  with a tiny double buffer. For sparse encoders (top- $\rho\%$ ) and token-selection decoders (top- $N_d$ ), a light gather-scatter unit remaps indices so external memory accesses remain burst-friendly; results carry query IDs so downstream order is preserved. The surrounding dense layers (pre-attention/projection and FFN) run on standard systolic GEMM engines.

In summary, DOOQ raises locality so fewer new lines are fetched; aligned prefetch hides most remaining latency; and the fused pass converts the locality into sustained utilization and end-to-end speedups. Section IV quantifies how  $w_d$  and  $r$  affect hit rate, stall time, and throughput across encoder/decoder and pruned settings, and attributes gains to improved hit rate  $h$ , regional reuse  $\rho$ , and fused-pass intensity.

#### IV. EVALUATION

We evaluate QUILL end-to-end and quantify how its schedule-aware prefetch and fused single-pass execution translate into system gains. We implement a synthesizable RTL generator in Chisel [34], verify with Verilator 5.009, and synthesize with Synopsys Design Compiler using a topographical flow [35] in TSMC 28nm HVT at 1GHz. External memory is modeled as HBM2 (256GB/s, 1.21pJ/bit [36]). Unless noted, we use COCO 2017 [37] and standard Deformable DETR settings ( $N_e = \lceil 20097 \cdot \rho \rceil$ ,  $N_d = 300$ ,  $D = 256$ ,  $M = 8$ ,  $L = 4$ ,  $K = 4$ ), consistent with common SOTA variants [11]–[16]. The region-prefetch radius  $r$  is fixed per level to the empirical maximum sampling offset from the model; we ablate only the DOOQ window  $w_d$ .

##### A. Ablation: DOOQ Lookup Window ( $w_d$ )

Fig. 9 sweeps  $w_d$  against a same-capacity direct-mapped baseline (no DOOQ logic) and reports encoder/decoder/total hit rate, memory energy efficiency, and area.

Encoder locality grows quickly with  $w_d$  (Fig. 9(a)) and exceeds **80%** by  $w_d = 1024$ —a **13.9×** improvement over baseline—because spatially adjacent queries reuse feature lines across levels once the window exposes them to the cache. Decoder locality (Fig. 9(b)) saturates at **55–63%** near  $w_d \approx 256$ : with  $N_d = 300$  candidates, the window already spans most near neighbors, and the residual accesses are inherently scattered by the token-selection policy. Total hit rate (Fig. 9(c)) tracks the encoder trend as the encoder dominates the query count.

Energy improves monotonically with  $w_d$  (Fig. 9(d)) as fewer new lines are fetched per step in Equation (2), while the time-multiplexed sorter keeps area growth predictable (Fig. 9(e)):  $O(w_d)$  comparators reused over  $O(\log^2 w_d)$  steps fit in the per-query slack ( $D/p_d$  cycles) without timing risk. Balancing hit, energy, and area, we choose  $w_d = 512$  for the rest of the study: up to **10.0×** hit-rate gain vs. baseline, **1.2–3.1×** memory-energy improvement, and **< 0.42 mm<sup>2</sup>** overhead—an attractive energy return per mm<sup>2</sup>.

##### B. End-to-End Latency and Throughput

Fig. 10 shows how locality gains turn into system-level speedups. Moving from  $\rho = 0.5$  to  $\rho = 0.1$  reduces end-to-end latency by **4.06×** (Fig. 10(a)), close to the ideal **4.48×** dictated by query counts (10,349  $\rightarrow$  2,310), and clearly exceeding the GPU’s scaling baseline (cf. Fig. 3). The small gap at  $\rho = 0.5$  versus dense is consistent with the mid-sparsity hit-rate dip: encoder requests still reuse well, while decoder misses begin to dominate.

Per-query behavior (Fig. 10(b)) matches this: dense encoder queries are fastest; sparsity tightens decoder locality and lowers median per-query latency, yet some off-region accesses remain that the local reordering in Equation (2) cannot fully hide—explaining the residual gap to ideal scaling.

Throughput scales accordingly. Overall throughput (Fig. 10(c)) reaches **3.67×** at  $\rho = 0.1$  vs.  $\rho = 0.5$ , near the ideal **4.48×**. Query throughput (Fig. 10(d)) is highest for the dense encoder—up to **3.31×** over sparse decoder

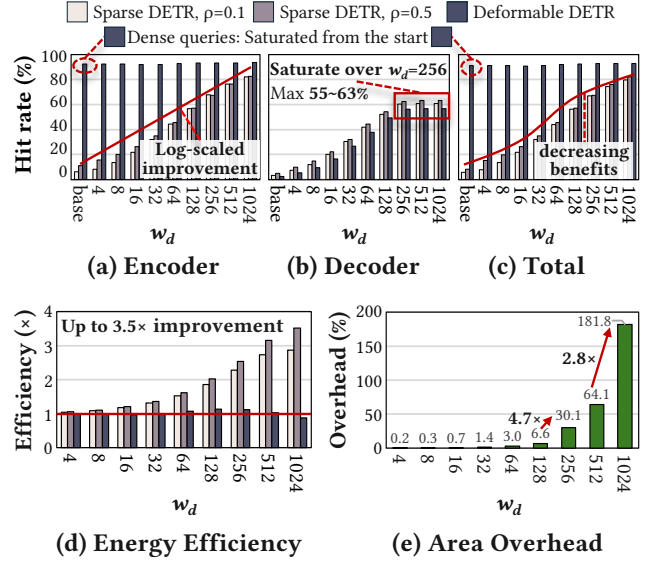


Fig. 9: DOOQ sweep over  $w_d$ . Encoder/decoder/total hit rate, memory energy efficiency, and area vs. a same-capacity direct-mapped cache.

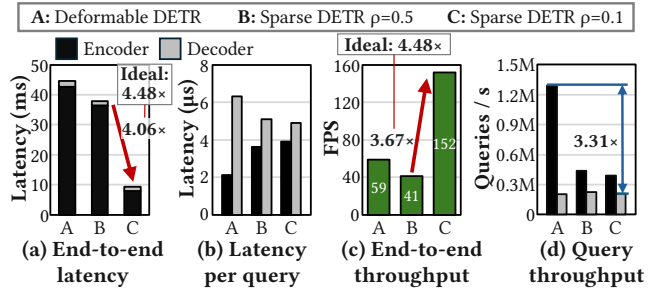


Fig. 10: End-to-end performance. (a) Overall latency. (b) Per-query latency. (c) Overall throughput. (d) Query throughput. Sparsity scales near-ideally ( $\rho = 0.5 \rightarrow 0.1$ ).

queries—thanks to stronger reuse and more uniform access patterns. In practice, aggressive pruning ( $\rho = 0.1$ ) delivers not only fewer FLOPs but also sustained wall-clock speedups.

##### C. Area and Power Breakdown

We next quantify the silicon budget of QUILL. Table I maps blocks to Fig. 6; gather-scatter ( $\textcircled{D}$ ) is folded into *Others*.

Area is memory-centric: on-chip SRAMs ( $\textcircled{C}$ ) plus the feature cache ( $\textcircled{A}$ ) occupy **7.60 mm<sup>2</sup>** ( $\approx 70\%$ ). Power is compute-centric: the two GEMMs ( $\textcircled{E}$ ) draw **49%**, reflecting FFN/projections. The fused MSDeformAttn core ( $\textcircled{B}$ ) is compact (1.35% area, 6.39% power), while DOOQ control (scheduler/sorter) is small—**3.83%** area and **2.92%** power—so most of the “DOOQ budget” is the cache capacity that enables the hit-rate gains in Fig. 9. For smaller footprints, first right-size the output SRAM (tiling/streaming), then tune  $w_d$ ;  $r$  is fixed by the model’s offset envelope, and the selector scales near-linearly, fitting in the  $D/p_d$  slack. With  $w_d = 512$ , the cache path adds **< 0.42 mm<sup>2</sup>** yet yields up to **10×** higher hit rate and 1.2–3.1× memory-energy improvement.

##### D. Model Accuracy (Baseline vs. QUILL)

We evaluate AP under FP32 and under QUILL—our fused single-pass fixed-point path with *mixed precision* (W8A8  $\rightarrow$  A18

TABLE I: Hardware evaluation at 1 GHz, 0.81 V (TSMC 28 nm HVT). Markers (Ⓐ–Ⓔ) follow Fig. 6. “Others” includes gather–scatter (Ⓓ) and clk/host interface.

Block	Area (mm <sup>2</sup> )	Area (%)	Power (mW)	Power (%)
DOOQ scheduler/sorter (Ⓐ)	0.416	3.83	83.4	2.92
Feature cache L1–L4 (Ⓐ)	1.306	12.04	241.6	8.44
Fused MSDeformAttn core (Ⓑ)	0.146	1.35	182.7	6.39
GEMM engines (32×32+64×64) (Ⓒ)	1.919	17.69	1406.0	49.15
On-chip SRAMs (other) (Ⓒ)	6.297	58.04	788.6	27.56
Others (incl. Ⓓ)	0.766	7.06	158.6	5.54
<b>Total</b>	<b>10.85</b>	<b>100</b>	<b>2860.9</b>	<b>100</b>

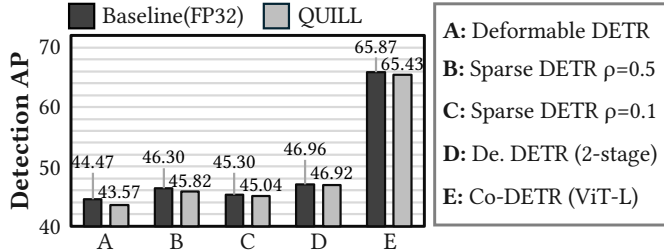


Fig. 11: **Detection accuracy (AP): FP32 vs. QUILL.** A: Deformable DETR; B: Sparse DETR ( $\rho=0.5$ ); C: Sparse DETR ( $\rho=0.1$ ); D: 2-stage; E: Co-DETR (ViT-L). QUILL is within  $\leq 0.9$  AP of FP32.

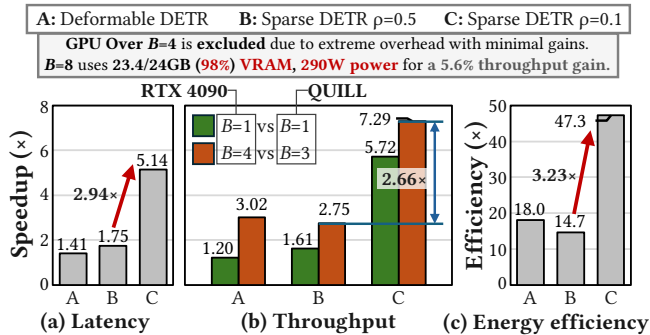


Fig. 12: **GPU comparison (RTX 4090).** (a) Latency. (b) Throughput vs. batch size. (c)  $B=1$  energy efficiency.

for linear/projection and bilinear, W16A8→A28 for attention-weighted aggregation/Softmax) [38]. Model semantics are unchanged except DOOQ scheduling. Figure 11 covers Deformable DETR, Sparse DETR at  $\rho=0.5/0.1$ , a two-stage variant, and large Co-DETR (ViT-L). Across all cases, QUILL is within **0.04–0.90** AP of FP32, preserving accuracy while enabling the locality-driven speedups above.

### E. Cross-Platform Analyses

We benchmark QUILL against a high-end GPU and recent MSDeformAttn accelerators.

**vs. RTX 4090.** Fig. 12 compares latency, throughput vs. batch size ( $B$ ), and  $B=1$  energy. QUILL delivers up to **5.14**× lower latency, **7.29**× higher throughput, and **47.3**× better  $B=1$  energy efficiency, with the largest margins at  $\rho=0.1$ . Batch scaling is favorable: we exceed the GPU at  $B=3$  (GPU needs  $B=4$ ), avoiding large- $B$  VRAM/power overheads.

**vs. Dedicated accelerators.** Table II places QUILL alongside DEFA [20] and UEDA [21]. Unlike DEFA (encoder-only, micro-kernelized, INT12) and UEDA (FPGA, reduced MSDA), QUILL accelerates *MSDeformAttn end-to-end with FFN* in a single fused pass while preserving FP32 semantics. This scope plus DOOQ-driven locality yields the **highest peak through-**

TABLE II: **Comparison with other accelerators**, including not only SOTA MSDeformAttn (MSDA) accelerators.

Metric	Unit	This work QUILL	DAC’24 DEFA [20]	ASPDAC’25 UEDA [21]
Workload	-	MSDA, FFN	MSDA (Encoder only)	MSDA (Shrunken*)
Precision	-	W8A8→A18, W16A8→A28	INT12	Not specified
Technology	-	28 nm	40 nm	FPGA
Area	mm <sup>2</sup>	10.85	2.63	-
Power	mW	2860.9	99.8	3070.0
Frequency	MHz	1000	400	200
Peak Throughput	TOPS	<b>12.06</b>	0.42	0.68
Power Efficiency	TOPS / W	<b>4.23</b>	4.19	0.22
Area Efficiency	TOPS / mm <sup>2</sup>	<b>1.11</b>	0.16	-
Normalized* Power Efficiency	TOPS / W	<b>4.23</b>	<b>6.84</b>	-
Normalized* Area Efficiency	TOPS / mm <sup>2</sup>	<b>1.11</b>	0.28	-
Normalized* Throughput	FPS	<b>1,270.5<sup>a</sup>–3832.7<sup>b</sup></b>	390.3	108.0
Utilization* Efficiency	FPS / TOPS	<b>31.76<sup>a</sup>–95.82<sup>b</sup></b>	9.76	2.70
Normalized* Energy Efficiency	FPS / W	<b>134.3<sup>a</sup>–405.1<sup>b</sup></b>	66.7	-

<sup>a</sup>Deformable DETR. <sup>b</sup>Sparse DETR  $\rho=0.1$ . \*Normalized to 28 nm process using the industry scaling model in [39].

**put** (12.06 TOPS) and **area efficiency** (1.11 TOPS/mm<sup>2</sup>) with competitive **power efficiency** (4.23 TOPS/W). On normalized end-to-end metrics, QUILL sustains **31.76–95.82 FPS/TOPS** and **134.3–405.1 FPS/W** (Deformable DETR → Sparse DETR,  $\rho=0.1$ ). Versus DEFA, throughput utilization and energy efficiency are **3.26–9.82**× and **2.01–6.07**× higher; UEDA trails due to FPGA frequency/resource limits and reduced configs. Overall, converting deformable attention into cache-friendly, single-pass work lets QUILL translate sparsity into end-to-end gains on real models while staying within  $\leq 0.9$  AP of FP32.

## V. CONCLUSION

We presented QUILL, an algorithm–architecture co-design that turns MSDeformAttn into cache-friendly, single-pass work. By coupling DOOQ with a fused MSDeformAttn core and integrated GEMMs, QUILL delivers near-linear sparse scaling and end-to-end gains: up to **7.29**× higher throughput and **47.3**× better energy efficiency than RTX 4090, and up to **9.82**×/6.07× higher throughput/energy efficiency than prior accelerators.

Three directions remain: (i) adaptive scheduling/prefetch (learned, multi-step) that tunes  $w_d$  and cache policy online; (ii) applying the schedule-aware prefetch loop to other sparse ops (cross-attention, video, point clouds) and to larger backbones via streamed  $W_m''$  tiling; and (iii) a compiler/runtime that auto-tunes micro-architectural knobs (parallelism  $p_d$ , GEMM tiling, cache partitioning) with tighter quantization-aware training (QAT) for mixed precision. Together, these extend QUILL’s core idea—turn sparsity into locality, then utilization—to the next wave of detection and multimodal transformers.

## ACKNOWLEDGEMENTS

This work was supported in part by the DARPA Young Faculty Award, the National Science Foundation (NSF) under Grants #2127780, #2319198, #2321840, #2312517, and #2235472, #2431561, the Semiconductor Research Corporation (SRC), the Office of Naval Research through the Young Investigator Program Award, and Grants #N00014-21-1-2225 and #N00014-22-1-2067, Army Research Office Grant #W911NF2410360. Additionally, support was provided by the Air Force Office of Scientific Research under Award #FA9550-22-1-0253, along with generous gifts from Xilinx and Cisco.

## REFERENCES

- [1] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," in *16th Eur. Conf. Comput. Vis. (ECCV)*, Glasgow, United Kingdom, Aug. 2020, p. 213–229.
- [2] Y. Zhao *et al.*, "DETRs Beat YOLOs on Real-time Object Detection," in *IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Seattle, WA, USA, Jun. 2024, pp. 16 965–16 974.
- [3] X. Gu, H. Fan, Y. Huang, T. Luo, and L. Zhang, "Context-Guided Spatio-Temporal Video Grounding," in *IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Seattle, WA, USA, Jun. 2024, pp. 18 330–18 339.
- [4] A. Kamath, M. Singh, Y. LeCun, G. Synnaeve, I. Misra, and N. Carion, "MDETR - Modulated Detection for End-to-End Multi-Modal Understanding," in *IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Montreal, QC, Canada, Oct. 2021, pp. 1760–1770.
- [5] J. Xu *et al.*, "Pixel Aligned Language Models," in *IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Seattle, WA, USA, Jun. 2024, pp. 13 030–13 039.
- [6] W. Huang *et al.*, "Tell Me What to Track: Infusing Robust Language Guidance for Enhanced Referring Multi-Object Tracking," *arXiv preprint arXiv:2412.12561*, 2024. [Online]. Available: <https://arxiv.org/abs/2412.12561>
- [7] S. Jeong, H. E. Barkam, H. Oh, H. Chen, T. Das, Z. Ye, and M. Imani, "iTaskSense: Task-Oriented Object Detection in Resource-Constrained Environments," in *62nd Annu. Des. Autom. Conf. (DAC)*, San Francisco, CA, USA, 2025, pp. 1–7.
- [8] X. Zhu, W. Su, L. Lu, B. Li, X. Wang, and J. Dai, "Deformable DETR: Deformable Transformers for End-to-End Object Detection," in *Int. Conf. Learn. Represent. (ICLR)*, Virtual, May 2021, pp. 1–16.
- [9] X. Zhu, H. Hu, S. Lin, and J. Dai, "Deformable ConvNets V2: More Deformable, Better Results," in *IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Long Beach, CA, USA, Jun. 2019, pp. 9300–9308.
- [10] Z. Zou, K. Chen, Z. Shi, Y. Guo, and J. Ye, "Object Detection in 20 Years: A Survey," *Proc. IEEE*, vol. 111, no. 3, pp. 257–276, 2023.
- [11] Z. Zong, G. Song, and Y. Liu, "DETRs with Collaborative Hybrid Assignments Training," in *IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Paris, France, Oct. 2023, pp. 6748–6758.
- [12] H. Zhang *et al.*, "DINO: DETR with Improved DeNoising Anchor Boxes for End-to-End Object Detection," in *Int. Conf. Learn. Represent. (ICLR)*, Kigali, Rwanda, May 2023, pp. 1–19.
- [13] F. Li, H. Zhang, S. Liu, J. Guo, L. M. Ni, and L. Zhang, "DN-DETR: Accelerate DETR Training by Introducing Query Denoising," in *IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, New Orleans, LA, USA, Jun. 2022, pp. 13 619–13 627.
- [14] B. Roh, J. Shin, W. Shin, and S. Kim, "Sparse DETR: Efficient End-to-End Object Detection with Learnable Sparsity," in *Int. Conf. Learn. Represent. (ICLR)*, Virtual, Apr. 2022, pp. 1–23.
- [15] S. Liu *et al.*, "Detection transformer with stable matching," in *IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Vancouver, BC, Canada, Jun. 2023, pp. 6491–6500.
- [16] C.-B. Zhang, Y. Zhong, and K. Han, "Mr. DETR: Instructive Multi-Route Training for Detection Transformers," in *IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Nashville, TN, USA, Jun. 2025, p. to appear.
- [17] J. Liang *et al.*, "Recurrent Video Restoration Transformer with Guided Deformable Attention," in *Adv. Neural Inf. Process. Syst. 35 (NeurIPS)*, New Orleans, LA, USA, 2022, pp. 378–393.
- [18] W. Wang *et al.*, "VisionLLM: Large Language Model is also an Open-Ended Decoder for Vision-Centric Tasks," in *Adv. Neural Inf. Process. Syst. 36 (NeurIPS)*, New Orleans, LA, USA, 2023, pp. 61 501–61 513.
- [19] P. Dong *et al.*, "SpeedDETR: Speed-aware Transformers for End-to-end Object Detection," in *40th Int. Conf. Mach. Learn. (ICML)*, Honolulu, Hawaii, USA, 2023.
- [20] Y. Xu *et al.*, "DEFA: Efficient Deformable Attention Acceleration via Pruning-Assisted Grid-Sampling and Multi-Scale Parallel Processing," in *61st Annu. Des. Autom. Conf. (DAC)*, San Francisco, CA, USA, Jun. 2024.
- [21] K. Sun, M. Wang, J. Zhou, and Z. Wang, "UEDA: A Universal And Efficient Deformable Attention Accelerator For Various Vision Tasks," in *30th Asia S. Pac. Des. Autom. Conf. (ASP-DAC)*, Tokyo, Japan, Jan. 2025, pp. 163–169.
- [22] T. Liang and G. Zeng, "FSH-DETR: An Efficient End-to-End Fire Smoke and Human Detection Based on a Deformable DETection TRansformer (DETR)," *Sensors*, vol. 24, no. 13, 2024.
- [23] R. Child, S. Gray, A. Radford, and I. Sutskever, "Generating Long Sequences with Sparse Transformers," *arXiv preprint arXiv:1904.10509*, 2019. [Online]. Available: <http://arxiv.org/abs/1904.10509>
- [24] J. Zhuang *et al.*, "SSR: Spatial Sequential Hybrid Architecture for Latency Throughput Tradeoff in Transformer Acceleration," in *ACM/SIGDA Int. Symp. Field Program. Gate Arrays (FPGA)*, Monterey, CA, USA, 2024, p. 55–66.
- [25] S. Tuli and N. K. Jha, "AccelTran: A Sparsity-Aware Accelerator for Dynamic Inference With Transformers," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 42, no. 11, pp. 4038–4051, 2023.
- [26] Q. Guo, J. Wan, S. Xu, M. Li, and Y. Wang, "HG-PIPE: Vision Transformer Acceleration with Hybrid-Grained Pipeline," in *IEEE/ACM Int. Conf. Comput. Aided Des. (ICCAD)*, Newark, NJ, USA, Oct. 2024, p. to appear.
- [27] C. Chen, L. Li, and M. M. Sabry Aly, "ViTA: A Highly Efficient Dataflow and Architecture for Vision Transformers," in *Des. Autom. Test Eur. (DATE)*, Valencia, Spain, 2024, pp. 1–6.
- [28] W. Li, Y. Luo, and S. Yu, "RAWatten: Reconfigurable Accelerator for Window Attention in Hierarchical Vision Transformers," in *Des. Autom. Test Eur. (DATE)*, Antwerp, Belgium, 2023, pp. 1–6.
- [29] E. Kwon, H. Song, J. Park, and S. Kang, "Mobile Accelerator Exploiting Sparsity of Multi-Heads, Lines, and Blocks in Transformers in Computer Vision," in *Des. Autom. Test Eur. (DATE)*, Antwerp, Belgium, 2023, pp. 1–6.
- [30] Y. Zhai, B. Li, B. Yan, and J. Wang, "STAR: An Efficient Softmax Engine for Attention Model with RRAM Crossbar," in *Des. Autom. Test Eur. (DATE)*, Antwerp, Belgium, 2023, pp. 1–2.
- [31] H. W. Oh, H. Chen, S. Yun, Y. Ni, B. Khaleghi, F. Wen, and M. Imani, "A Multimodal AI Acceleration with Dynamic Pruning and Run-Time Configuration," in *IEEE 33rd Annu. Int. Symp. Field-Program. Cust. Comput. Mach. (FCCM)*, Fayetteville, AR, USA, 2025, pp. 293–293.
- [32] H. W. Oh, J. Park, and S. E. Lee, "DL-Sort: A Hybrid Approach to Scalable Hardware-Accelerated Fully-Streaming Sorting," *IEEE Trans. Circuits Syst. II*, vol. 71, no. 5, pp. 2549–2553, May 2024.
- [33] M. E. Sadeghi, A. Fayyazi, S. Azizi, and M. Pedram, "PEANO-ViT: Power-Efficient Approximations of Non-Linearities in Vision Transformers," in *29th ACM/IEEE Int. Symp. Low Power Electron. Des. (ISLPED)*, Newport Beach, CA, USA, Aug. 2024, pp. 1–6.
- [34] J. Bachrach *et al.*, "Chisel: constructing hardware in a Scala embedded language," in *49th Annu. Des. Autom. Conf. (DAC)*, San Francisco, CA, USA, Jun. 2012, pp. 1216–1225.
- [35] "DC Ultra: Concurrent Timing, Area, Power, and Test Optimization," Synopsys, Tech. Rep., 2018. [Online]. Available: <https://www.synopsys.com/content/dam/synopsys/implementation%26signoff/datasheets/dc-ultra-ds.pdf>
- [36] S. Ghodrati, H. Sharma, C. Young, N. S. Kim, and H. Esmaeilzadeh, "Bit-Parallel Vector Composability for Neural Acceleration," in *57th Annu. Des. Autom. Conf. (DAC)*, Virtual, Jul. 2020, pp. 1–6.
- [37] T.-Y. Lin *et al.*, "Microsoft COCO: Common Objects in Context," in *13th Eur. Conf. Comput. Vis. (ECCV)*, Zurich, Switzerland, 2014, pp. 740–755.
- [38] H. W. Oh, S. An, W. S. Jeong, and S. E. Lee, "RF2P: A Lightweight RISC Processor Optimized for Rapid Migration from IEEE-754 to Posit," in *ACM/IEEE Int. Symp. Low Power Electron. Des. (ISLPED)*, Vienna, Austria, 2023, pp. 1–6.
- [39] W. Huang, K. Rajamani, M. R. Stan, and K. Skadron, "Scaling with Design Constraints: Predicting the Future of Big Chips," *IEEE Micro*, vol. 31, no. 4, pp. 16–29, Jul. 2011.