
Article:
Liu, S., Zhao, Y., Liu, H. et al. (2026) An energy-efficient FPGA-based transformer accelerator for AIoT consumer electronics using dual-side diagonal sparsity. IEEE Transactions on Consumer Electronics. ISSN: 0098-3063

https://doi.org/10.1109/tce.2026.3688916

An Energy-Efficient FPGA-Based Transformer Accelerator for AIoT Consumer Electronics Using Dual-Side Diagonal Sparsity

Sichen Liu, Yuqin Zhao, Haotian Liu, Haihang Xia, Xinyi Wang, John Goodenough, Charith Abhayaratne, *Member, IEEE,* and Tiantai Deng

***Abstract*—Consumer electronics, such as smartphones, wearables, and smart cameras, increasingly demand real-time and energy-efficient on-device intelligence enabled by Artificial Intelligence (AI) and the Internet of Things (IoT) to support perception and control under strict latency and power constraints. Transformer-based models have been increasingly adopted in sensing and perception tasks due to their ability to capture contextual relationships via self-attention. However, their high computational and memory costs limit their practicality on resource- and power-constrained Artificial Intelligence of Things (AIoT) devices. To address these challenges, we propose a dual-side diagonal sparsity mechanism that partitions the Q and K matrices into 2×2 blocks and applies diagonal pruning within each block, reducing computation and memory overhead while preserving transposition invariance and hardware-friendly regularity. We design the Sparse Processor Architecture for Transformer Acceleration (SPATA), a Field-Programmable Gate Array (FPGA)-based framework for energy-efficient on-device Transformer inference on AIoT devices, supporting both sparse–sparse and dense–dense matrix multiplications in multi-head self-attention (MHSA). SPATA integrates a Dual-Mode Matrix Unit (DMMU) along with a diagonal prune–compress module and a vector unit supporting fused addition and rectified linear unit (ReLU) operations. Experimental results on a Xilinx Kintex-7 FPGA show that SPATA achieves 168.48 giga operations per second (GOP/s) throughput, 0.05 ms latency, and 82.18 GOP/J energy efficiency on MHSA workloads, delivering up to a 40× speedup over Central Processing Units (CPUs) and Graphics Processing Units (GPUs) baselines and outperforming prior FPGA-based Transformer accelerators.**

***Index Terms*—Consumer electronics, Transformer acceleration, on-device inference, energy-efficient FPGA design, AIoT.**

I. Introduction

ALWAYS-ON and interactive consumer electronics applications impose strict constraints on latency, power consumption, and memory capacity. Such systems are widely deployed in consumer products, exemplified by smart cameras that perform continuous visual sensing and low-latency inference for tasks such as object detection and event recognition under limited energy budgets. To satisfy these stringent requirements, smart cameras rely on local data processing and on-device inference rather than cloud-based computation, in order to achieve real-time responsiveness and reduce communication overhead [1], [2], [3]. Such design challenges are shared by a broader class of resource-constrained consumer electronics platforms, commonly referred to as the Artificial Intelligence Internet of Things (AIoT), where edge devices execute local inference under tight computational and memory constraints [4]. In this context, Transformer-based models are increasingly adopted in vision-centric AIoT pipelines, including smart cameras, to support perception-intensive workloads that require modelling long-range dependencies and global contextual information. However, when deployed in such resource-constrained consumer electronics platforms, the heavy reliance of Transformers on large-scale matrix multiplications and multi-head self-attention (MHSA) leads to rapidly increasing computational and memory demands, which have emerged as a critical bottleneck for achieving real-time and energy-efficient on-device deployment in AIoT consumer electronics [5], [6].

Vision Transformer (ViT) has demonstrated significant potential in AIoT-oriented perception tasks. By leveraging the self-attention mechanism, ViT can capture long-range dependencies and global contextual relationships, achieving superior performance in visual tasks such as image recognition and object detection [7], [8]. These capabilities make ViT an attractive backbone for perception in AIoT consumer electronics, including smart cameras, autonomous driving modules, AR glasses for real-time visual understanding, and smart speakers employing Transformer-based speech models. Recent studies have further demonstrated the effectiveness of Transformer-based architectures in consumer-oriented sensing and perception applications under real-world deployment constraints. In addition, AI-driven sensing frameworks are increasingly adopted in large-scale AIoT infrastructures, such as environmental monitoring and rainfall retrieval using commercial microwave links in 6G-IoT networks [9]. However, the high computational complexity and redundant operations introduced by patch-based tokenization in ViT models significantly increase latency and energy consumption, which severely limit their real-time deployment on resource- and power-constrained edge devices. Therefore, improving the computational efficiency of ViT models and reducing redundant operations are essential for enabling practical on-device deployment in AIoT environments and consumer electronics systems under strict latency and energy budgets [10], [11].

Building upon the foundation established by the ViT, re-

Sichen Liu, Yuqin Zhao, Haotian Liu, Haihang Xia, Xinyi Wang, John Goodenough, and Charith Abhayaratne are with the School of Electrical and Electronic Engineering, The University of Sheffield, S1 3JD Sheffield, U.K.

Tiantai Deng is with the Donghai Lab, Zhoushan, China. (e-mail: dengtiantai@donghailab.com).

Sichen Liu, Yuqin Zhao, and Haotian Liu contributed equally to this work.

search has focused on improving its computational efficiency and deployment feasibility on resource-constrained platforms. Although ViT achieves global feature modelling capability through self-attention, it faces challenges in computational efficiency and limited local feature modelling due to the absence of convolutional inductive bias. To overcome these limitations, Convolutional Neural Network (CNN)–Transformer hybrid architectures have been proposed, particularly for vision tasks in consumer and edge applications, to combine the complementary strengths of CNN and ViT [12]. In these architectures, convolutional layers extract local spatial features, while Transformer layers capture long-range dependencies [13], [14]. Recent investigations have advanced hybrid CNN–Transformer modelling by incorporating the dynamics of Ordinary Differential Equations (ODEs) into CNN–Transformer frameworks [15], [16]. By replacing early self-attention layers with ODE-based modules, these models interpret feature evolution as a continuous process, reducing redundant attention computations while improving parameter efficiency and training stability. As a result, ODE-based CNN–Transformer architectures achieve a better trade-off between accuracy and efficiency, making them attractive for perception-intensive consumer electronics such as smart cameras.

In the CNN–Transformer architecture enhanced with Neural ODEs, the ODE-based modules in the early stages capture the continuous temporal and spatial evolution of features, leading to smoother information propagation and reduced redundant transformations [17]. To further enhance representational capacity and maintain accuracy, MHSA layers are typically introduced after the ODE modules, as MHSA is essential for modelling global contextual dependencies and long-range interactions [18]. However, despite its effectiveness, MHSA incurs substantial computational and memory overhead, since its complexity scales quadratically with the sequence length and relies heavily on large-scale matrix multiplications [19]. Such characteristics lead to increased latency, energy consumption, and memory access pressure, which pose significant challenges for deployment on resource- and power-constrained AIoT consumer devices. As a result, MHSA becomes a dominant performance and energy bottleneck in practical on-device inference systems. Therefore, reducing the computational cost and improving the hardware efficiency of MHSA is a critical requirement for enabling real-time and energy-efficient Transformer inference in consumer electronics platforms [20].

To mitigate the quadratic complexity of MHSA, various sparsity-based methods have been explored to prune redundant attention connections. Random sparsity removes a portion of attention elements during inference, but its unstructured pattern causes irregular memory access and low hardware utilization [21]. Structured *N:M* sparsity introduces partial regularity by retaining N non-zero elements in every group of M elements (e.g., 2:4), enabling mapping to specialized hardware [22]. However, it remains sensitive to matrix alignment and transposition, which can reduce effective sparsity and hardware efficiency in attention computation [23]. Edge-aware sparsity preserves salient local regions while pruning homogeneous areas, but often generates irregular patterns and overlooks long-range dependencies, making efficient hardware acceleration difficult [24]. As a result, sparsity schemes without sufficient structural regularity and hardware awareness suffer from inefficient data access, low PE utilization, and limited energy-efficiency gains on FPGA-based and embedded accelerators [25].

A review of sparsity-driven optimization for attention mechanisms reveals several inherent limitations that hinder the performance and energy efficiency of AI applications on resource-constrained edge devices. One of the primary reasons lies in the mismatch between algorithmic sparsity patterns and hardware execution regularity, which leads to inefficient dataflow scheduling and underutilized computation units. The key challenges can therefore be summarized as follows:

1) Mainstream sparsity schemes (e.g., *N:M* sparsity) suffer from transposition dependence and alignment waste, which degrades hardware regularity.
2) Existing bitmap and Compressed Sparse Row (CSR)-based encodings incur substantial indexing and storage overhead and lead to irregular memory access patterns.
3) Existing FPGA-based Transformer accelerators lack reconfigurable frameworks and lightweight compute units to efficiently support heterogeneous sparse and dense attention workloads, limiting real-time performance and energy efficiency.

To overcome the above limitations of existing sparsity schemes, we propose a diagonal sparsity pattern that partitions matrices into 2×2 blocks and distributes non-zero elements diagonally within each block, ensuring structural regularity and cross-dimensional consistency, along with its corresponding FPGA implementations. The proposed pattern eliminates alignment waste, preserves transposition invariance, and offers superior hardware friendliness, reducing computational and memory access overhead without compromising model accuracy. In contrast to random sparsity, which suffers from dimension dependency upon transposition [21], the proposed diagonal sparsity pattern preserves structural consistency and maintains a uniform sparsity distribution regardless of matrix transposition. Compared with edge-based sparsity [26], it preserves both local and partial long-range dependencies while avoiding unstructured irregular sparsity patterns. This diagonal distribution aligns naturally with the wavefront propagation direction in systolic arrays, enabling continuous data flow and higher multiply–accumulate (MAC) utilization efficiency. The detailed contributions are as follows:

1) We propose a dual-side diagonal sparsity mechanism that preserves transposition invariance, avoids alignment waste, and ensures consistent sparsity for the query–key ($QK^\top$) and relative-position (R) products $QR^\top$. Compared with the 2:4 pattern, it achieves higher accuracy while retaining hardware regularity.
2) We further propose a diagonal compression scheme that leverages the diagonal sparsity structure to reduce indexing overhead to one quarter of traditional bitmap formats. This scheme achieves up to $1.88\times$ compression, improves on-chip memory bandwidth utilization, and enables efficient sparse-matrix storage and retrieval.
3) We propose SPATA (Sparse Processor Architecture for

Transformer Acceleration), a reconfigurable FPGA architecture that supports both sparse–sparse and dense–dense attention computations. Implemented on a Xilinx XC7K325T FPGA, SPATA achieves 168.48 GOP/s throughput, 0.05 ms latency, and 82.18 GOP/J energy efficiency, delivering 9.2×–324.8× speedup over CPU/GPU baselines and up to 2.4× speedup over prior FPGA designs.

The remainder of this paper is organized as follows. Section II reviews the techniques used in this work. Section III introduces the proposed dual-side diagonal sparsity. Section IV describes the hardware architecture. Section V presents the experimental results, and Section VI concludes the paper.

II. Related Works

A. Vision Transformer

Transformer-based architectures have become a major research focus in computer vision. ViT applies self-attention to visual tasks and achieves strong performance in image recognition and other vision applications [27], [28]. Transformer models have also been applied in other domains, including communication and sensing systems [29]. Compared with CNNs that primarily capture local spatial structures, Transformers are more effective at modelling long-range contextual dependencies. However, Transformers typically require large datasets and substantial computational resources for training, and the quadratic complexity of MHSA with respect to the input size N leads to high computational and memory costs [30], [31]. Consequently, achieving real-time and energy-efficient ViT inference on resource-constrained devices remains a significant challenge [32].

The primary computational bottleneck of ViTs lies in the MHSA module. In each attention head, the Q, K, and value (V) feature maps are first generated through three linear projections, followed by two intensive matrix multiplications: $QK^\top$ for attention score computation and the subsequent multiplication with V for weighted feature aggregation. These matrix multiplications scale quadratically with the sequence length, resulting in dominant contributions to both the overall floating-point operations (FLOPs) and memory bandwidth consumption. As a result, optimizing the efficiency of the Q, K, and V matrix multiplications within MHSA has become a key direction for accelerating ViT inference, particularly on hardware-constrained platforms [33].

B. CNN–Transformer Model

To reduce the computational overhead of pure Transformer models and enhance model performance on limited data, researchers have proposed hybrid architectures that integrate convolutional and attention mechanisms [34]. By combining the respective strengths of CNNs and Transformers, these hybrid models improve feature expressiveness and data efficiency, enabling high accuracy even on small- to medium-sized datasets across various vision tasks. Among them, the Bottleneck Transformer Network (BoTNet) replaces the spatial convolutions in the last three bottleneck blocks of ResNet with

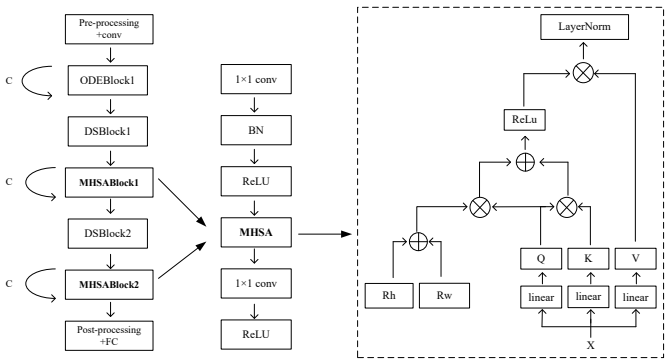

Fig. 1. CNN–Transformer architecture with Neural ODE blocks and the corresponding MHSA computation module.

global self-attention modules, significantly improving both image classification and object detection accuracy [12]. This demonstrates the effectiveness of embedding MHSA blocks into convolutional backbones.

In parallel, Neural ODE–based networks such as dsODENet reformulate residual networks as continuous-depth models governed by differential equations [35], [36]. By replacing multiple residual blocks with a single ODE block and reusing parameters across integration steps, these models reduce parameter redundancy and improve training stability. CNN–Transformer hybrid architectures combine convolution and attention mechanisms to enhance feature representation while modelling both local spatial features and global contextual dependencies [37]. This design philosophy also motivates the CNN–Transformer architecture with Neural ODE integration illustrated in Fig. 1.

C. Sparse Transformer Acceleration

Despite their strong performance, Transformer models incur high computational and memory costs due to massive parameters and matrix multiplications, making sparsity an important strategy for improving hardware efficiency. Early studies on unstructured sparsity pruned redundant weights but produced irregular data patterns that degraded hardware utilization [21]. In contrast, structured *N:M* sparsity (e.g., 2:4) provides hardware-friendly regularity while maintaining accuracy at moderate sparsity levels. Recent hardware-aware approaches further exploit both weight and activation sparsity using techniques such as sparse Tensor Cores, sparse convolution, and sparse general matrix multiplication (SpGEMM), enabling more efficient inference on modern accelerators. Moreover, sparse attention mechanisms reduce the quadratic complexity of self-attention by computing only selected Q–K pairs, improving scalability for large models and long-sequence tasks.

However, despite its efficiency advantages, sparsity also introduces several inherent limitations. As illustrated in Fig. 2, when the number of columns is not divisible by four, row-wise 2:4 sparsity must either pad extra zeros (upper illustration) or discard the remaining elements (lower illustration) to maintain

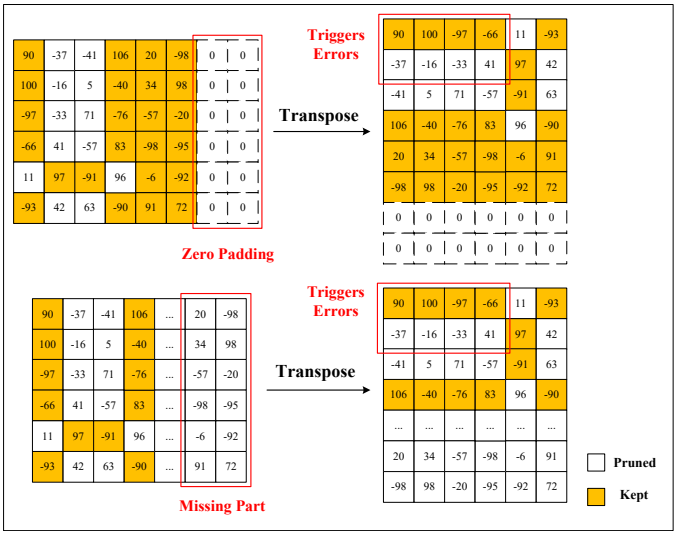

Fig. 2. Limitations of 2:4 structured sparsity. Irregular matrix shapes lead to grouping misalignment (left), and transposition destroys the 4-element grouping, resulting in the loss of structured sparsity (right).

four-element grouping before transposition. After transposition, the original row-wise 2:4 grouping is no longer preserved, because zero padding or missing tail elements shift the four-element blocks and cause the previously aligned non-zero entries to become scattered and irregular. Such irregularity leads to uneven data distribution and irregular memory access, which further disrupts hardware scheduling. In addition, the indexing and mask management required to represent sparse matrices incur extra storage and control overhead, which can offset the theoretical computational savings when sparsity is moderate. Therefore, designing storage formats and hardware architectures that preserve structured sparsity while minimizing control overhead remains a critical challenge in achieving practical sparse Transformer acceleration [38].

D. Sparse Storage Format Optimization

Despite being fundamental to scientific computing and machine learning accelerators, existing SpGEMM implementations still face major efficiency bottlenecks. Inner-product-based approaches waste time fetching rare non-zero elements, while outer-product-based designs suffer from excessive partial matrix generation and inefficient result merging [39]. To address these challenges, several techniques have been proposed to enhance dataflow efficiency. A highly parallel, streaming-based merger pipelines the multiplication and merging stages, allowing partial results to be combined on-chip promptly after generation [40]. A condensed matrix representation further reduces partial matrices and minimizes DRAM access, while a Huffman tree–based scheduler improves the scalability of the merger for large sparse matrices. In addition, a row prefetcher with a near-optimal cache replacement policy ensures balanced input access and reduced latency, benefiting from consistent row-major data formatting [41].

To further reduce memory footprint and improve data reuse, various sparse storage formats have been explored. The Coordinate (COO) format directly stores non-zero element tuples, offering simple implementation but large storage overhead, while the Compressed Sparse Row (CSR) format provides better compression but introduces irregular indexing and control complexity. In FPGA-based accelerators, data reordering strategies have been proposed to improve hardware utilization and reduce memory access overhead [41]. However, CSR remains inefficient for highly parallel implementations due to memory fragmentation and redundant accesses. To address these limitations, optimized formats such as Cyclic Channel Sparse Row (C2SR) and Variable Block Row (VBR) have been proposed to improve bandwidth efficiency and computational throughput [42], [43]. In addition, scheduling techniques such as the extended Hu algorithm reorder non-zero elements to generate contiguous data streams, enabling efficient streaming computation on hardware accelerators [44].

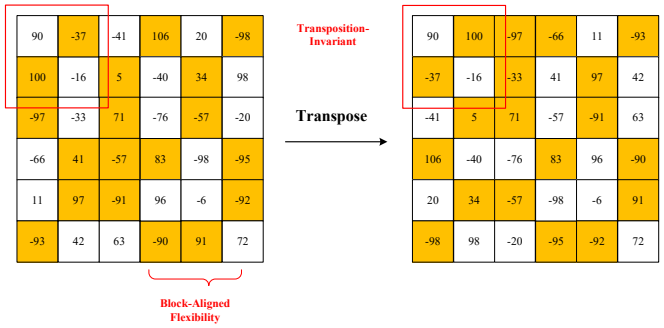

Fig. 3. Advantages of dual-side diagonal sparsity. The pattern is preserved after transposition, and the diagonal structure supports block-aligned pruning.

III. Proposed Dual-side Diagonal Sparsity Mechanism

A. Advantages of Dual-Side Diagonal Sparsity

To overcome the limitations of traditional 2:4 structured sparsity, we introduce a dual-side diagonal sparsity mechanism that eliminates row-end alignment waste and preserves structural regularity under transposition. The mechanism partitions the matrix into 2×2 blocks and retains diagonal elements within each block, where the diagonal selection of each block is uniquely determined by a combination of row-side and column-side masks, without rearranging the original matrix entries. As illustrated in Fig. 3, the highlighted region shows an example of a retained diagonal group: yellow cells indicate elements that lie on the selected diagonal, whereas white cells correspond to pruned positions. Because diagonal grouping is performed at the 2×2 block granularity, it only requires even row and column sizes, offering a finer-grained constraint than the strict four-element divisibility imposed by traditional 2:4 sparsity, and thereby avoiding unprunable leftover elements at row ends. Moreover, diagonal structures remain diagonal after transposition, making the sparsity pattern inherently transposition-invariant and hardware-friendly, which is particularly beneficial for CNN–Transformer workloads with frequent matrix transpositions in multi-head attention.

The dual-side diagonal sparsity offers several key advantages for hardware acceleration. First, pruning along both

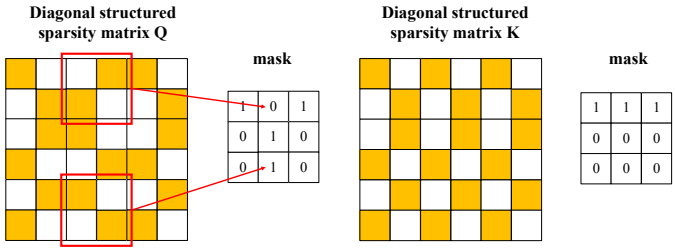

Fig. 4. Positional encoding under the proposed dual-side diagonal sparsity.

row and column dimensions eliminates misaligned fragments and forms compact diagonal groups that map uniformly onto systolic arrays, thereby improving computational efficiency and dataflow regularity compared with single-side sparsity, which enables continuous data streaming, sustains high MAC utilization across PEs, and significantly reduces pipeline stalls and idle cycles during execution. Second, the pattern is highly predictable, as the diagonal structure is preserved under both multiplication orientations and remains unchanged after transposition, which simplifies indexing, stabilizes sparsity ratios across $QK^\top$ and $QR^\top$, and enables more regular memory-access behaviour for CNN–Transformer workloads. Finally, from an algorithmic perspective, the structured diagonal connections preserve both local and partial long-range dependencies, maintaining contextual information more effectively than unstructured, edge-based, or *N:M* sparsity patterns, which often disrupt structural consistency and weaken important cross-dimensional correlations.

B. Dual-Side Diagonal Selection Mechanism

Within each 2×2 block, the diagonal sparsity pattern is encoded into a binary mask that specifies which diagonal orientation is retained. As illustrated in Fig. 4, blocks that preserve the main diagonal (top-left and bottom-right elements) are assigned a mask value of 1, whereas blocks that retain the anti-diagonal (top-right and bottom-left elements) are assigned a value of 0. This diagonal-to-bit transformation converts each local sparse block into a single-bit structural descriptor, producing a compact and hardware-friendly representation of the underlying sparsity. In contrast to traditional fine-grained schemes (e.g., unstructured sparsity or *N:M* sparsity) that must store explicit coordinate information or multi-bit index patterns, the proposed diagonal mask requires only one bit per block, substantially lowering metadata overhead and enabling deterministic, low-cost hardware lookup. For the Q and K matrices, this encoding yields a streamlined diagonal mask array in which each bit directly corresponds to the diagonal orientation preserved in its associated 2×2 block, thus simplifying address generation, control logic, and downstream datapath alignment.

After the diagonal masks of Q and K are independently generated, they are spatially aligned and combined through a dual-side XNOR operation to determine the effective diagonal orientation assigned to each 2×2 block in the output. When the row-side and column-side masks encode the same diagonal orientation, the XNOR result is 1, indicating that the main diagonal contributes to the computation; when the two encodings differ, the result is 0, indicating that the anti-diagonal orientation is adopted instead. This process harmonizes the diagonal choices derived from both matrices and produces a coherent diagonal-selection map that governs how each block contributes to the resulting attention matrix. By determining diagonal orientation rather than altering sparsity density, the mechanism preserves the global 50% sparsity while ensuring that the retained computation exhibits consistent structural properties. The resulting selection map enables the output matrix to maintain a regular diagonal pattern across blocks, which is particularly advantageous for hardware scheduling and pipeline alignment. Moreover, because diagonal structures remain stable under transposition, the resulting pattern naturally inherits transposition invariance, making it especially well suited for CNN–Transformer workloads that frequently rely on transpose operations in multi-head attention.

C. Dual-Side Diagonal Parameter Compression

Compared with the traditional bitmap-based 2:4 sparse storage format, the proposed dual-side diagonal sparsity mechanism offers substantial improvements in both storage overhead and access efficiency. Let the parameter matrix be $W \in \mathbb{R}^{R\times C}$,and assume each element is quantized to q bits.

For the bitmap representation, half of the elements remain non-zero, leading to a value storage requirement of

$$\text{Value bits} = q \times \frac{RC}{2}, \tag{1}$$

and since one mask bit is required for each element, the mask storage overhead is

$$\text{Mask bits} = RC. \tag{2}$$

Thus, the total storage cost becomes

$$\text{Total bits}_{\text{bitmap}} = q\frac{RC}{2} + RC. \tag{3}$$

In contrast, the proposed dual-side diagonal sparsity mechanism exploits the structured placement of non-zero elements along diagonals, requiring only a small amount of positional metadata. The resulting total storage is

$$\text{Total bits}_{\text{diag}} = q\frac{RC}{2} + \epsilon RC, \quad \epsilon = \frac{1}{4}, \tag{4}$$

where ϵ denotes the lightweight indexing overhead introduced by the diagonal encoding.

Relative to the dense storage cost qRC, the compression ratios are

$$\text{CR}_{\text{bitmap}} = \frac{q\frac{RC}{2} + RC}{qRC} = \frac{q/2+1}{q}, \tag{5}$$

$$\text{CR}_{\text{diag}} = \frac{q\frac{RC}{2} + \epsilon RC}{qRC} = \frac{q/2+\epsilon}{q}. \tag{6}$$

For an 8-bit quantization ($q = 8$), these evaluate to, where the compression factor is defined as $1/\text{CR}$,

$$\text{CR}_{\text{bitmap}} = \frac{4+1}{8} = 0.625 \quad (\text{i.e., } 1.6\times \text{ compression}), \tag{7}$$

$$\mathrm{CR}_{\mathrm{diag}} = \frac{4 + 0.25}{8} = 0.53125 \quad (\text{i.e., } 1.88\times \text{ compression}). \tag{8}$$

This demonstrates that our dual-side diagonal sparsity mechanism compression significantly reduces positional overhead compared to bitmap, achieving better overall compression while maintaining the same value storage. Additionally, the structured pattern simplifies decoding and access logic, reducing memory bandwidth demands, which is especially beneficial for hardware inference deployment.

IV. Architecture

A. Overall Architecture

This paper introduces a sparsity-aware FPGA architecture designed to efficiently accelerate MHSA in CNN–Transformer models by combining dual-side diagonal sparsity with a unified computation unit that supports both sparse–sparse and dense–dense matrix multiplications. To enable efficient sparse–sparse computation within this unified framework, we introduce a novel dual-side matrix multiplication algorithm that explicitly leverages structured sparsity in both the Q and K matrices. This algorithm is further supported by a custom compressed sparse array format that reduces memory footprint and improves data access efficiency. To mitigate memory-bandwidth bottlenecks, the architecture employs hardware-friendly storage formats and dataflow scheduling strategies that enhance on-chip data reuse and reduce off-chip transfers. Together, these designs enable efficient sparse MHSA inference on low-power FPGA platforms, reducing redundant computation and energy consumption while supporting real-time edge intelligence.

Building upon this design, Fig. 5 presents the overall architecture of SPATA, which is organized into three functional subsystems: memory, compute, and control. On the memory side, the weight memory and input memory supply model parameters and activations, while the temporary memory stores temporary results produced during MHSA computation. At the core of the compute subsystem, the DMMU performs sparse–sparse and dense–dense matrix multiplications using a systolic array of SDMMUs, and its outputs are further processed by the vector unit and Layer Normalization (LayerNorm) module to complete addition, residual, activation, and normalization operations. The entire dataflow is orchestrated by the control subsystem: the top control unit coordinates global timing, while the address control, compute control, and LayerNorm control modules independently manage memory addressing, matrix-unit execution, and normalization sequencing. Together, these components form a tightly integrated architecture that supports efficient sparse MHSA execution under strict on-chip resource and bandwidth constraints.

Overall, the dual-side sparsity algorithm, unified DMMU architecture, and compressed sparse array format form a co-optimized pipeline that minimizes redundant computation, reduces memory traffic, and increases systolic reuse. The algorithm exposes regular diagonal patterns, allowing zero-valued operations to be skipped in a structured manner, while the DMMU maps them onto its dual-mode sparse–sparse and dense–dense compute architecture with lightweight mask

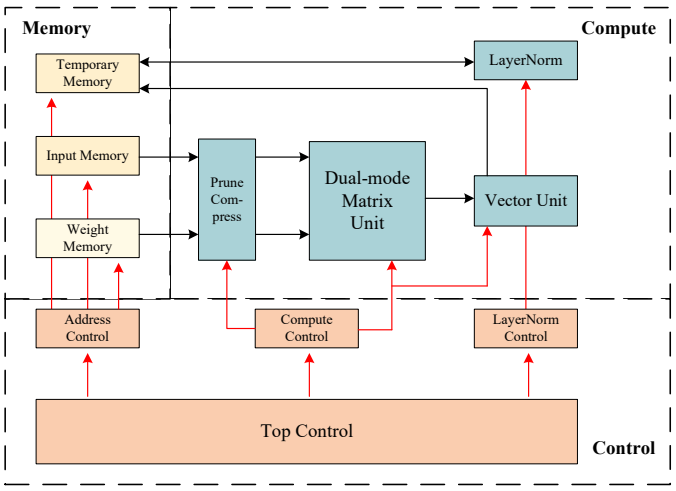

Fig. 5. Overall architecture of SPATA. Black arrows denote dataflow, while red arrows indicate control signalling. Yellow blocks correspond to memory modules, orange blocks represent control logic, and blue blocks denote the compute units.

control and direct datapath reuse. Meanwhile, the compressed format ensures bandwidth-efficient data delivery, which further lowers memory access overhead and supports continuous operand feeding into the systolic array. Owing to the regular diagonal structure, data propagation is naturally aligned with the systolic wavefront, reducing pipeline bubbles and simplifying sparse scheduling, thereby lowering latency. At the same time, the unified DMMU reuses the same PE datapath across sparse and dense workloads, and the compact 1-bit diagonal encoding reduces the hardware cost of metadata storage, indexing, and mask control, leading to improved area efficiency. Together, this algorithm–architecture co-design allows SPATA to sustain high throughput under stringent FPGA resource constraints, improve energy efficiency through reduced computation and data movement, and enable real-time sparse MHSA acceleration on low-power edge platforms.

B. DMMU Architecture

The DMMU employs a 9×9 array of SDMMUs to provide unified acceleration for both sparse–sparse and dense–dense matrix multiplications in CNN–Transformer models. In dense mode, the array processes full input vectors and projection weights to support operations such as $Q \times W_Q$ and $K \times W_K$. For attention computations such as $Q \times K^\top$ and $Q \times R^\top$, the array operates in sparse mode, where compressed sparse representations and binary masks enable the selective activation of valid multiply–accumulate operations, thereby reducing the computational workload. Each SDMMU delivers a fixed-latency pipelined output that is forwarded to a shared buffer for downstream attention processing.

To support these two computation modes within a single hardware unit, each SDMMU dynamically reconfigures the connection between its two internal PEs. The PEs can be chained to form a two-stage pipeline for dense linear projections, allowing a single input vector to propagate through

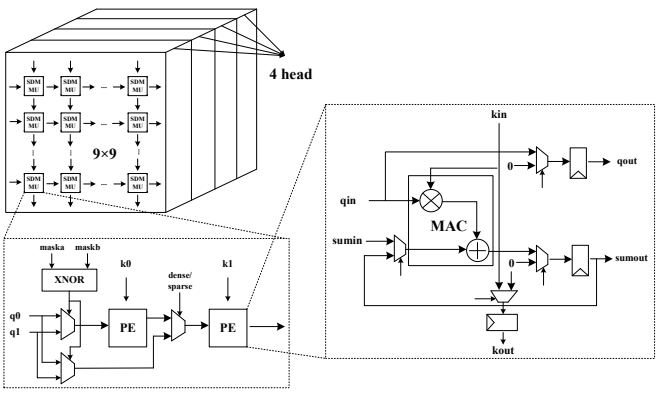

Fig. 6. Hierarchical architecture of the DMMU. It features four parallel matrix-multiplication units, each containing a 9×9×2 unified systolic PE array capable of both sparse–sparse and dense–dense computation.

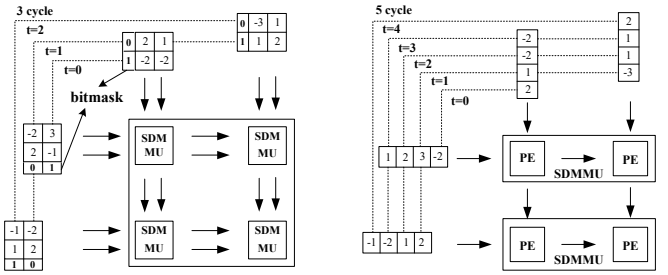

Fig. 7. Sparse and dense dataflows of the DMMU. In the sparse mode (left), binary masks (bitmasks) filter the valid inputs before entering the SDMMU. In the dense mode (right), all elements are processed directly by the PE units.

both stages before being merged into a unified result. In sparse mode, the two PEs operate fully independently, with each PE receiving its own sparse input vector and mask and computing only on valid nonzero entries. This dual-PE reconfiguration enables high utilization while avoiding redundant zero-valued operations, providing an efficient microarchitectural foundation for both projection and attention workloads.

As shown in Fig. 6, the hierarchical organization of the DMMU is illustrated across three levels. The left portion presents the 9×9 systolic array replicated across four attention heads, enabling parallel multi-head processing through wavefront-style data propagation. The bottom-left inset highlights the structural composition of an individual SDMMU, providing its internal datapath, including the dual-PE structure, mask-selection logic, input/weight forwarding paths, and the configurable sparse-dense execution circuitry. Specifically, during sparse computation, the masks from the Q and K matrices are combined via an XNOR operation within the mask-selection logic to determine the effective diagonal alignment for each 2×2 block, which further controls, through datapath multiplexers, the selection of predefined data pairs (as produced by the Prune Compress module and organized by the memory layout), thereby determining which elements are fetched and fed into the PE for multiplication. The right-side inset details the SDMMU's MAC pipeline, including the multiplier, accumulator, bypass paths, and forwarding logic for q_{out}, k_{out}, and sum_{out}. Together, these views demonstrate how the systolic array, SDMMU datapath, and MAC pipeline collectively support efficient sparse–sparse and dense–dense matrix multiplications within CNN–Transformer workloads.

C. Supporting Efficient Matrix Multiplication

As shown in Fig. 7, the left subfigure illustrates how the DMMU accelerates sparse matrix multiplication by skipping invalid positions indicated by the binary mask, allowing only valid nonzero entries to be forwarded to the MAC units. This selective computation reduces the workload and shortens the execution to three cycles per block. In contrast, the right subfigure shows the dense matrix multiplication dataflow, where all input elements are processed without masking. In this mode, the two internal PEs of the SDMMU form a two-stage pipeline, enabling full dense computation that completes in five cycles per block. Together, the two dataflows demonstrate how the DMMU adapts its execution pattern under sparse and dense configurations to maximize computational efficiency.

Both sparse and dense matrix-multiplication workloads in our architecture are executed using a unified strategy that integrates head-level parallelism with loop-based data traversal to maximize hardware utilization. For dense computations, such as $Q \times W_Q$ and $K \times W_K$, each head independently processes its own projection using a dedicated set of dense weight matrices, while loop iterations over input partitions enable pipelined execution and efficient reuse of compute resources. In sparse computations, including $Q \times K^\top$ and $Q \times R^\top$, each head handles a portion of the sparse attention computation, with loops iterating over compressed matrix blocks and their corresponding binary masks. This allows the SDMMU array to selectively perform multiply–accumulate operations only on valid nonzero elements, significantly reducing redundant work. By integrating head-based parallelism with structured loop scheduling, the architecture provides a scalable and flexible mechanism for efficiently executing both sparse and dense workloads in CNN–Transformer models.

As shown in Fig. 8, the DMMU adopts different data access patterns for sparse–sparse and dense–dense matrix multiplications to maximize computation efficiency. In the dense case, both the input matrix and the weight matrix are partitioned into head-level tiles, where each attention head fetches its corresponding dense weight block while input activations are streamed into the DMMU through loop-based traversal. This enables independent per-head processing and fully parallel dense matrix multiplication execution with a systolic dataflow. In contrast, the sparse mode redesigns the access path to accommodate the compressed K and R matrices and their binary masks. Rather than loading full dense tiles, each head retrieves only the valid nonzero elements and structural indices from the sparse matrix, while the Q matrix is streamed iteratively into the DMMU. This selective access mechanism skips invalid positions and performs multiply–accumulate operations only where needed, reducing both memory bandwidth usage and redundant computation. Together, these data access strategies allow the DMMU to support both sparse and dense execution

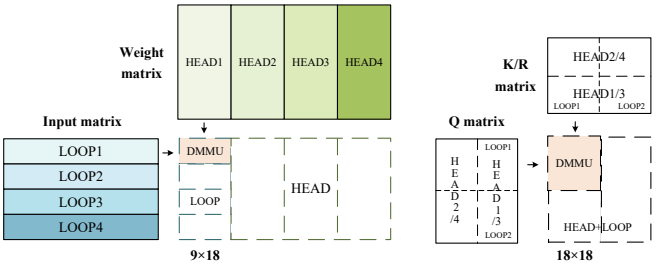

Fig. 8. Dense and sparse data-access patterns of the DMMU. Dense mode (left) streams 9×18 input tiles into the DMMU, whereas sparse mode (right) loads 18×18 blocks from the dual-side diagonally Q and K/R matrices.

modes through adaptive data mapping and streaming.

D. Prune Compress and Vector Unit Modules

The Prune Compress Module consists of 32 lightweight submodules operating in parallel, each handling a distinct segment of the input vector. In each cycle, every submodule processes one 2×2 block and uses a 1-bit mask to select either the main diagonal or anti-diagonal pair. The mask directly controls deterministic data-path selection based on the row-major memory layout, avoiding explicit address computation. Collectively, the 32 submodules extract 32 valid element pairs (64 elements) with positional metadata per cycle, and over two consecutive cycles these outputs are concatenated into a compact 2×64 sparse vector. Meanwhile, each cycle generates a 32-bit mask describing the structural pattern of the processed blocks; these masks are concatenated across cycles to form a 1024-bit structured mask row and stored in temporary memory for subsequent sparse attention operations such as $Q \times K^\top$ and $Q \times R^\top$. This parallel compression pipeline establishes a direct mapping from the 1-bit structural descriptor to memory-side data selection, enabling efficient hardware realization of the sparsity mechanism and allowing the SDMMU array to operate directly on compressed representations with reduced computation and memory bandwidth.

The Vector Unit is organized as an array of processing elements designed for high-throughput post-processing of SDMMU outputs. It performs activation functions (e.g., ReLU), element-wise accumulation, and lightweight vector transformations required by subsequent layers. Through parallel execution across the PE array, the module processes large vector segments concurrently, improving throughput and minimizing latency. Activation operations enhance nonlinear expressiveness, while accumulation supports the fusion of intermediate results across heads or computation stages. As a bridge between matrix multiplication and downstream network components, the Vector Unit ensures efficient and real-time vector-level processing within the CNN–Transformer pipeline.

V. Experimental Results

A. Experimental Setup

1) STL10 and CIFAR100 Datasets: In this work, we evaluate our method on two widely used image classification benchmarks, STL10 and CIFAR100, which represent different levels of dataset scale and visual complexity. CIFAR100 contains 60,000 images across 100 categories with a resolution of 32×32, favouring models with strong local inductive biases such as CNNs, and prior studies [12] show that CNN–Transformer hybrids also perform well by combining local and global feature modelling. In contrast, STL10 provides higher-resolution 96×96 images with richer spatial structures, including 10 labelled categories and a large unlabelled subset, making it suitable for evaluating architectures that rely on long-range dependency modelling such as self-attention. Using both datasets allows us to assess the effectiveness of the proposed dual-side diagonal sparsity across small-scale, locally dominated settings and higher-resolution, globally correlated visual tasks, providing a comprehensive evaluation of its behaviour within CNN–Transformer hybrid architectures.

2) On-board Execution: The proposed accelerator is deployed on a Xilinx Kintex-7 XC7K325T FPGA to evaluate its actual hardware performance. The design is synthesized and implemented using the Xilinx Vivado 2020.2 toolchain. The accelerator is implemented at the register-transfer level (RTL) using Verilog hardware description language (HDL). All compute modules, including the 9×9×4 SDMMU array, the dual-side diagonal Prune Compress unit, and the vector post-processing unit, operate under a unified clock domain after timing closure. The Q, K, and R feature maps and model parameters are preloaded into on-board BRAM through a host initialization step, and the complete execution is carried out entirely on the FPGA without CPU intervention.

During run time, the on-chip controller sequentially streams the preloaded data into the compute pipeline, enabling wavefront-style propagation across the SDMMU array, while all intermediate results are accumulated locally and written back to the temporary buffer for subsequent layers. The reported latency, cycle counts, and throughput are collected through on-chip hardware performance counters during real on-board execution on the XC7K325T device, and therefore reflect the end-to-end execution behaviour of the deployed accelerator. For power evaluation, we use Vivado post-implementation power analysis based on the implemented design after place-and-route. The switching activity is generated from workload-driven RTL simulation and annotated to the post-implementation design, so that both static and dynamic power under the target MHSA workload are included. Accordingly, the reported energy efficiency is derived from the on-board measured throughput and the post-implementation power result.

B. Algorithmic Optimizations

1) Classification Accuracy: On the STL-10 dataset, the ODE–ODE–MHSA architecture, following the baseline design in [15], achieves an accuracy of 75.57%. In comparison, a three-layer MHSA stack (MHSA–MHSA–MHSA) reaches 79.09%. Replacing one MHSA layer with an ODE layer yields the hybrid ODE–MHSA–MHSA model, which further improves accuracy to 79.66%, as summarized in Table I. These results indicate that MHSA layers are essential for

TABLE I
ACCURACY COMPARISON ON STL-10 DATASET

Model Structure	Accuracy (%)
ODE – ODE – MHSA	75.57
MHSA – MHSA – MHSA	79.09
ODE – MHSA – MHSA	79.66

TABLE II
ACCURACY COMPARISON ON STL-10 DATASET UNDER SPARSITY STRATEGIES, INCLUDING CANNY-BASED SPARSITY [45] AND *N:M* STRUCTURED SPARSITY [46].

Algorithm	Sparsity pattern		Accuracy (%)
	Attention	Q and K	
No algorithm	✗	✗	79.66
Canny sparsity	✓	✗	69.94
	✗	✓	63.45
N:M sparsity	✓	✗	79.45
	✗	✓	73.31
Our algorithm	✓	✗	76.50
	✗	✓	74.17
	✓	✓	69.63

feature modelling and that ODE layers provide complementary benefits. However, the quadratic complexity of MHSA introduces significant computational and memory overhead, limiting efficiency and scalability. To address this issue, the proposed dual-side diagonal sparsity preserves essential local and partial long-range dependencies while substantially reducing attention computation, thereby enabling efficient inference without compromising accuracy and making the approach suitable for resource-constrained environments and large-scale vision workloads.

As shown in Table II, the model achieves 79.66% accuracy on STL-10 under the non-sparse setting. When Canny-based sparsification is applied to the attention matrix, accuracy drops sharply to 69.94% because this method preserves only edge regions while discarding a large amount of non-edge and long-range information. This degradation becomes more severe, with accuracy dropping to 63.45% when both Q and K are sparsified. In comparison, the *N:M* sparsity scheme maintains 79.45% accuracy when sparsifying only the attention matrix, indicating that fixed-ratio pruning better preserves structural integrity. However, its performance declines markedly to 63.31% when both Q and K are sparsified due to its randomness and lack of structural regularity, which hinder consistent coverage of local and partial long-range dependencies. In contrast, the proposed dual-side diagonal sparsity achieves 76.50% accuracy when sparsifying attention, outperforming Canny and approaching the performance of *N:M*, while offering a more structured and hardware-friendly pattern that effectively preserves essential local relationships. Moreover, when both Q and K are sparsified, it still reaches 69.63%, outperforming both Canny and *N:M* sparsity, demonstrating its superior ability to retain essential local and partial long-range information while mitigating sparsity-induced degradation.

We observe that the performance trends identified on STL-10 are consistently reproduced on CIFAR-100. As shown in Table III, Canny-based sparsity causes substantial accuracy degradation in both datasets. Likewise, *N:M* sparsity maintains high accuracy when applied solely to attention but exhibits notable drops when sparsifying both Q and K. In contrast, the proposed dual-side diagonal sparsity achieves a more favourable balance between preserving essential local and partial global information and reducing computational cost. The consistency of these results across two datasets with different scales and characteristics demonstrates that our approach generalizes well and is not dependent on dataset-specific artifacts or coincidental effects.

TABLE III
ACCURACY COMPARISON ON CIFAR-100 DATASET UNDER SPARSITY STRATEGIES, INCLUDING CANNY-BASED SPARSITY [45] AND *N:M* STRUCTURED SPARSITY [46].

Algorithm	Sparsity pattern		Accuracy (%)
	Attention	Q and K	
No algorithm	✗	✗	71.81
Canny sparsity	✓	✗	57.40
	✗	✓	54.46
N:M sparsity	✓	✗	70.50
	✗	✓	65.76
Our algorithm	✓	✗	67.36
	✗	✓	66.89
	✓	✓	62.20

2) Error Analysis: We compare the error distributions of the traditional 2:4 sparsity algorithm and the proposed dual-side diagonal sparsity under matrix multiplication with normally distributed inputs, as illustrated in Fig. 9. Both methods exhibit uniform error distributions without noticeable local accumulation, indicating that errors are evenly distributed across the $Q \times K$ matrix and that computational stability is preserved. Although the dual-side diagonal sparsity constrains non-zero elements along fixed diagonal positions, its error behaviour remains consistent with that of the 2:4 pattern, with no observable structural bias.

Further quantitative analysis shows that the error distribution of the proposed method follows a near-normal pattern, with a slightly lower peak near zero and marginally larger deviations in the 0.25–0.75 range compared to 2:4 sparsity; however, these differences remain small and within acceptable bounds for practical applications. By accepting an accuracy reduction of approximately 3%, the proposed method enables faster computation and improved storage efficiency, achieving a favourable trade-off between numerical precision and computational efficiency.

To support these observations with objective metrics, we evaluate the Mean Squared Error (MSE) and Signal-to-Quantization-Noise Ratio (SQNR) of the $Q \times K^\top$ computation under both sparsity methods, averaged over 20 trials with normally distributed inputs across head dimensions of 64, 128, and 256 and sequence lengths of 64, 128, and 256. The proposed dual-side diagonal sparsity achieves an average normalized MSE of 28.7 and an SQNR of 3.51 dB, compared to 15.8 and 6.08 dB for 2:4 structured sparsity. Although lower due to the stricter structural constraint, the SQNR of the proposed method remains stable within ± 0.03 dB across all

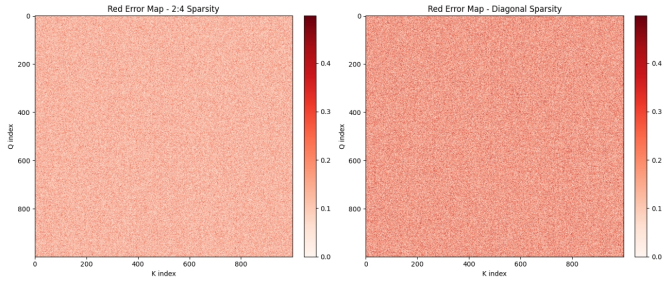

Fig. 9. Element-wise error distributions of the 2:4 structured sparsity (left) and the proposed dual-side diagonal sparsity (right).

configurations, indicating the absence of dimension-dependent degradation. This numerical consistency, together with the classification results in Table II, confirms that the introduced numerical trade-off does not lead to unacceptable task-level accuracy loss, thereby validating the claimed balance between numerical precision and hardware efficiency.

C. Compute-Unit Resource and Power Analysis

To quantify the hardware cost introduced by the proposed reconfigurable compute unit, Table IV compares a single SDMMU with a baseline SDMMU (BSDMMU) on the same Artix-7 A7-100T FPGA. Although the SDMMU incorporates dual-PE reconfiguration, mask-selection logic, and configurable sparse and dense execution support, it requires only 234 LUTs, 302 FFs, and 2 DSPs, compared with 377 LUTs, 446 FFs, and 4 DSPs for the BSDMMU. At nearly the same power level, the SDMMU also achieves a higher operating frequency, reaching 158 MHz at 0.123 W, compared with 156 MHz at 0.121 W for the BSDMMU. These results indicate that the proposed compute unit provides execution flexibility without introducing significant hardware or power overhead. Instead, the SDMMU achieves a substantially lower implementation cost while maintaining comparable power consumption and a higher operating frequency, demonstrating resource efficiency.

This advantage becomes pronounced at the array level. As shown in Table V, a 9×9 SDMMU array implemented on the ZCU102 uses only 19K LUTs, 27K FFs, and 162 DSPs, whereas the 9×9 baseline SDMMU (BSDMMU) array requires 27K LUTs, 44K FFs, and 324 DSPs. The proposed array also achieves lower power and a higher frequency, namely 0.683 W at 210 MHz, compared with 0.716 W at 200 MHz for the 9×9 BSDMMU array. This array-level advantage mainly arises from two factors. First, the SDMMU datapath is replicated across processing elements, so the resource savings of each compute unit are accumulated over the entire 9×9 array, leading to a larger reduction in LUT, FF, and DSP usage. Second, the proposed architecture preserves regular sparse execution and structured dataflow across the array, which reduces redundant switching activity, avoids duplicated dense computation resources, and improves hardware utilization. As a result, the proposed design not only maintains its unit-level efficiency after array integration, but further amplifies its advantages when inter-PE communication, sparse-control support, and systolic scheduling are considered together.

TABLE IV
UNIT-LEVEL RESOURCE AND POWER COMPARISON ON ARTIX-7 A7-100T

Design	LUTs	FFs	DSPs	Power (W)	Frequency (MHz)
SDMMU	234	302	2	0.123	158
BSDMMU	377	446	4	0.121	156

TABLE V
ARRAY-LEVEL RESOURCE AND POWER COMPARISON ON ZCU102

Design	LUTs	FFs	DSPs	Power (W)	Frequency (MHz)
9×9 SDMMU array	19K	27K	162	0.683	210
9×9 BSDMMU array	27K	44K	324	0.716	200

TABLE VI
KEY CONFIGURATIONS OF TRANSFORMER-BASED MODELS

Model	Encoders	Decoders	Sequence Length	Heads	Hidden Size	FFN Size
TinyBERT4	4	0	128	12	312	1200
Dino-vits8	12	0	128	6	384	1536
Transformer-base (Enc.)	6	0	64	8	512	2048
Transformer-base (Dec.)	0	6	64	8	512	2048
Shallow Transformer	2	1	64	4	200	800
CNN–Transformer	2	0	64	4	256	1024

TABLE VII
FPGA RESOURCE UTILIZATION OF THE SPATA ACCELERATOR

Platform	Frequency	LUTs	FFs	BRAMs	DSPs
SPATA (XC7K325T)	170 MHz	127K (62.56%)	134K (32.92%)	30 (6.74%)	162 (19.28%)

D. Hardware Resource Utilization

To contextualize the hardware resource utilization, the neural network configurations executed on the SPATA accelerator are introduced. The accelerator is primarily designed to support a CNN–Transformer model with larger hidden and intermediate dimensions than a representative Shallow Transformer, resulting in higher computational demands and enhanced representational capacity. Although the Shallow Transformer incorporates a decoder layer while the CNN–Transformer adopts an encoder-only structure, the two models share comparable encoder depth, sequence length, and attention-head configurations. The key parameters of all benchmark models are summarized in Table VI.

In light of the model configurations described above, we analyze the FPGA resource utilization of the proposed accelerator. Table VII summarizes the resource utilization of SPATA implemented on the Xilinx XC7K325T platform. Operating at 170 MHz, the design utilizes 62.56% (127K) of the available LUTs and 32.92% (134K) of the flip-flops, indicating balanced logic-resource usage. The BRAM utilization remains relatively low at 6.74% (30 blocks), reflecting efficient on-chip memory management. In addition, 162 DSP slices are employed, accounting for 19.28% of the available DSP resources and

TABLE VIII
COMPARISON OF THE PROPOSED ACCELERATOR WITH PRIOR CPU/GPU/FPGA DESIGNS

Platform	CPU	GPU			FPGA						
	i9-9900X	Jetson Nano	RTX 2080 Ti	RTX 3090	SOCC'20 [47]	ISLPED'20 [48]	ISQED'21 [49]	STA-Tiny'22 [50]	STA-Small'22 [50]	FNM-Trans'24 [46]	Our work
Chip	Skylake-X	Tegra X1	TU102	GA102	XCVU13P	XCVU9P	XCU200	XC7Z020	XC7VX485T	XCU200	XC7K325
Technology	14 nm++	20 nm	12 nm	8 nm	16 nm	16 nm	16 nm	28 nm	28 nm	16 nm	28 nm
Frequency	3.50 GHz	640 MHz	1.35 GHz	1.70 GHz	200 MHz	-	-	150 MHz	200 MHz	200 MHz	170 MHz
Sparsity Scheme	-	-	-	2:4 group-based pruning	Low-bit quantization	Block-circulant matrix with FFT	Block-based pruning	*N*:*M* group-based pruning		2:8 weight & attention sparsity	Dual-side diagonal sparsity
LUT Utilization (KLUT)	-	-	-	-	1728	268	1882	21	116	558	127
DSP Utilization	-	-	-	-	12288	5647	6840	132	1040	4096	162
CLB Utilization (KCLB)	-	-	-	-	245	146	372	5	35	70	19
# MAC units	-	-	-	-	4096	∼5647	∼3368	128	1024	4096	648
Bit Precision	FP-32	FP-32	FP-32	FP-32	FIX-8	FIX-16	-	FIX-16	FIX-16	FIX-16	FIX-16
Latency (ms)	2.17	16.24	1.70	0.46	0.30	2.94	0.32	2.01	0.42	0.12	**0.05**
Batch-1 Throughput (GOP/s)	101.38	13.55	129.41	478.26	733.33	75.34	687.50	109.45	523.81	1833.33	**168.48**
Performance Density (GOP/s/KLUT)	-	-	-	-	0.42	0.28	0.37	5.21	4.51	4.93	1.33
Performance Density (GOP/s/DSP)	-	-	-	-	0.06	0.01	0.10	0.83	0.50	0.67	**1.04**
Performance Density (GOP/s/KCLB)	-	-	-	-	2.99	0.52	1.85	21.89	14.97	39.44	8.87
Power (W)	165.00	7.56	250.00	350.00	16.70	22.45	-	2.71	9.87	28.23	**2.05**
Energy Efficiency (GOP/J)	0.61	1.79	0.52	1.37	43.91	3.35	-	40.39	53.07	64.94	**82.18**

highlighting the compute-intensive nature of SPATA's processing engine. Overall, the resource footprint demonstrates that SPATA achieves an effective balance between performance and hardware efficiency, making it well suited for deployment on mid-range FPGA platforms.

E. Overall System Evaluation

Overall, SPATA demonstrates an efficient and well-balanced system design across hardware cost, performance, and energy consumption. As summarized in Table VIII, SPATA employs only 648 MAC units, substantially fewer than STA-Small and other prior FPGA accelerators, due to its dual-side diagonal sparsity architecture and optimized dataflow. Although implemented on a mid-range FPGA, SPATA achieves an ultra-low latency of 0.05 ms and a throughput of 168.48 giga operations per second (GOP/s), delivering over $40\times$ lower latency and significantly higher throughput than STA-Tiny under similar resource constraints. While its throughput is lower than that of STA-Small implemented on high-end devices, SPATA provides up to an $8\times$ latency advantage, making it particularly suitable for real-time edge applications. Compared with FNM-Trans, which is implemented on a high-end XCU200 FPGA with 4096 MAC units and adopts a 2:8 weight and attention sparsity scheme, SPATA achieves significantly lower latency (0.05 ms compared with 0.12 ms) while using substantially fewer computational resources. Moreover, SPATA attains an energy efficiency of 82.18 GOP/J with 2.05 W power consumption, surpassing SOCC'20 and STA-Small. These results collectively indicate that the proposed diagonal sparsity accelerator achieves an excellent trade-off between hardware efficiency, computational performance, and power consumption, positioning SPATA as a promising solution for low-power, real-time AI inference on resource-constrained FPGA platforms.

VI. CONCLUSION

This paper presents a dual-side diagonal sparsity mechanism and SPATA, a unified algorithm–hardware co-optimized framework for accelerating both sparse–sparse and dense–dense matrix multiplications on FPGAs under the resource and energy constraints of consumer electronics devices. The proposed diagonal sparsity preserves structural regularity, maintains transposition invariance, and effectively reduces indexing and memory access overhead. SPATA integrates a $9\times9\times4$ SDMMU array, a lightweight diagonal Prune Compress module, and a vector unit supporting fused addition and ReLU, providing a reconfigurable datapath that sustains high computational throughput for on-device Transformer inference. Experimental results on the Xilinx XC7K325T FPGA show that SPATA achieves 168.48 GOP/s throughput, 0.05 ms latency, and 82.18 GOP/J energy efficiency, outperforming representative CPU, GPU, and FPGA-based accelerators. These results validate the effectiveness of the proposed approach in enabling real-time, energy-efficient Transformer inference for consumer-oriented AIoT scenarios such as smart cameras and vision-enabled edge devices. Future work will investigate the scalability of the proposed framework to larger Transformer models with longer sequence lengths, more attention heads, and higher hidden dimensions. We will also further optimize FPGA resource allocation and scheduling strategies to improve support for multi-task and large-scale on-device inference in consumer electronics.

REFERENCES

[1] H. Zheng, Y. Zhao, B. Zhang, G. Shang, M. H. Yahya Al-Shamri, and H. Aldossary, "A low-resolution video action recognition approach based on multi-scale reconstruction and multi-modal fusion," *IEEE Transactions on Consumer Electronics*, vol. 71, no. 1, pp. 970–983, 2025.

[2] A. Hussain, S. U. Khan, N. Khan, M. W. Bhatt, A. Farouk, J. Bhola, and S. W. Baik, "A hybrid transformer framework for efficient activity recognition using consumer electronics," *IEEE Transactions on Consumer Electronics*, vol. 70, no. 4, pp. 6800–6807, 2024.

[3] L. Wang, Y. Shi, G. Mao, F. A. Dharejo, S. Javed, and M. Alathbah, "Consumer-centric insights into resilient small object detection: SCIoU loss and recursive transformer network," *IEEE Transactions on Consumer Electronics*, vol. 70, no. 1, pp. 2178–2187, 2024.

[4] O. Jouini, K. Sethom, A. Namoun, N. Aljohani, M. H. Alanazi, and M. N. Alanazi, "A survey of machine learning in edge computing: Techniques, frameworks, applications, issues, and research directions," *Technologies*, vol. 12, no. 6, p. 81, 2024.

[5] C. Hu and B. Li, "When the edge meets Transformers: Distributed inference with Transformer models," in *Proc. IEEE Int. Conf. Distributed Computing Systems (ICDCS)*, 2024, pp. 82–92.

[6] Krisvarish V, Priyadarshini T, K. P. Abhishek Sri Saai, and Vaidehi Vijayakumar, "Resource-efficient transformer architecture: optimizing memory and execution time for real-time applications," *arXiv preprint arXiv:2501.00042*, 2024.

[7] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16×16 words: Transformers for image recognition at scale," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2021, pp. 1–22.

[8] J. Chen, Y. Lu, Q. Yu, X. Luo, E. Adeli, Y. Wang, L. Lu, A. L. Yuille, and Y. Zhou, "TransUNet: Transformers make strong encoders for medical image segmentation," *arXiv preprint arXiv:2102.04306*, 2021.

[9] M. Wang, C. Han, K. Liu, F. Zhang, H. Chen, J. Huo, W. He, Y. Bi, Q. Feng, Z. Wang, and X. Li, "AI-enhanced rainfall retrieval using commercial microwave links in 6G-IoT networks: Advances, challenges, and opportunities," *IEEE Internet of Things Journal*, vol. 13, no. 5, pp. 7933–7951, Mar. 2026.

[10] S. Mehta and M. Rastegari, "MobileViT: Light-weight, general-purpose, and mobile-friendly Vision Transformer," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2022, pp. 1–14.

[11] Y. Rao, W. Zhao, B. Liu, J. Lu, J. Zhou, and C.-J. Hsieh, "DynamicViT: Efficient Vision Transformers with dynamic token sparsification," in *Advances in Neural Information Processing Systems*, vol. 34, 2021, pp. 13 937–13 949.

[12] A. Srinivas, T.-Y. Lin, N. Parmar, J. Shlens, P. Abbeel, and A. Vaswani, "Bottleneck Transformers for visual recognition," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2021, pp. 16 519–16 529.

[13] A. Khan, Z. Rauf, A. Sohail, A. R. Khan, H. Asif, A. Asif, and U. Farooq, "A survey of the vision transformers and their CNN-transformer based variants," *Artif. Intell. Rev.*, vol. 56, no. Suppl. 3, pp. 2917–2970, 2023.

[14] H. Wu, B. Xiao, N. Codella, M. Liu, X. Dai, L. Yuan, and L. Zhang, "CvT: Introducing convolutions to Vision Transformers," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, 2021, pp. 22–31.

[15] I. Okubo, K. Sugiura, and H. Matsutani, "A Cost-Efficient FPGA-Based CNN–Transformer using Neural ODE," *IEEE Access*, vol. 12, pp. 155 773–155 788, 2024.

[16] Y. D. Zhong, T. Zhang, A. Chakraborty, and B. Dey, "A neural ODE interpretation of Transformer layers," pp. 1–8, 2022, presented at the DLDE Workshop, NeurIPS 2022.

[17] S. Becker, M. Klein, A. Neitz, G. Parascandolo, and N. Kilbertus, "Predicting ordinary differential equations with Transformers," in *Proc. 40th Int. Conf. Mach. Learn. (ICML)*, 2023, pp. 1978–2002.

[18] B. Zhuang, J. Liu, Z. Pan, H. He, Y. Weng, and C. Shen, "A survey on efficient training of Transformers," in *Proc. Int. Joint Conf. Artif. Intell. (IJCAI)*, 2023, pp. 6823–6831.

[19] M. Farina, U. Ahmad, A. Taha, H. Younes, Y. Mesbah, X. Yu, and W. Pedrycz, "Sparsity in Transformers: A systematic literature review," *Neurocomputing*, vol. 582, p. 127468, 2024.

[20] X. Chen, H. Li, M. Li, and J. Pan, "Learning a sparse Transformer network for effective image deraining," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2023, pp. 5896–5905.

[21] T. Hoefler, D. Alistarh, T. Ben-Nun, N. Dryden, and A. Peste, "Sparsity in deep learning: pruning and growth for efficient inference and training in neural networks," *J. Mach. Learn. Res.*, vol. 22, no. 241, pp. 1–124, 2021.

[22] A. Zhou, Y. Ma, J. Zhu, J. Liu, Z. Zhang, K. Yuan, W. Sun, and H. Li, "Learning N:M fine-grained structured sparse neural networks from scratch," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2021, pp. 1–12.

[23] I. Hubara, B. Chmiel, M. Island, R. Banner, J. Naor, and D. Soudry, "Accelerated sparse neural training: A provable and efficient method to find N:M transposable masks," in *Advances in Neural Information Processing Systems*, vol. 34, 2021, pp. 21 099–21 111.

[24] Y. Feng, B. Han, X. Wang, J. Shen, X. Guan, and H. Ding, "Self-supervised Transformers for unsupervised SAR complex interference detection using Canny edge detector," *Remote Sensing*, vol. 16, no. 2, p. 306, 2024.

[25] C. Li, Y. Peng, G. Liu, Y. Li, X. Yang, and C. Chen, "Efficient vision transformer for human-centric AIoT applications through token tracking assignment," *IEEE Transactions on Consumer Electronics*, vol. 70, no. 1, pp. 1029–1039, 2024.

[26] I. Beltagy, M. E. Peters, and A. Cohan, "Longformer: The long-document transformer," *arXiv preprint arXiv:2004.05150*, 2020.

[27] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin Transformer: Hierarchical Vision Transformer using shifted windows," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, 2021, pp. 10 012–10 022.

[28] W. Wang, J. Dai, Z. Chen, Z. Huang, Z. Li, X. Zhu, X. Hu, T. Lu, L. Lu, H. Li, X. Wang, and Y. Qiao, "InternImage: Exploring large-scale vision foundation models with deformable convolutions," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2023, pp. 14 408–14 419.

[29] X. Li, Y. Gao, M. Zeng, X. Lei, W. Hao, A. Nallanathan, and O. A. Dobre, "Recent advances in resource allocation and beam prediction for large language models empowered ISAC systems," *IEEE Communications Magazine*, vol. 64, no. 4, pp. 113–119, Apr. 2026.

[30] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jégou, "Training data-efficient image Transformers & distillation through attention," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2021, pp. 10 347–10 357.

[31] T. Dao, "FlashAttention-2: Faster attention with better parallelism and work partitioning," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2024, pp. 1–21.

[32] B. Graham, A. El-Nouby, H. Touvron, P. Stock, A. Joulin, H. Jégou, and M. Douze, "LeViT: A vision transformer in ConvNet's clothing for faster inference," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, 2021, pp. 12 259–12 269.

[33] S. Wang, B. Li, M. Khabsa, H. Fang, and H. Ma, "Linformer: Self-attention with linear complexity," *arXiv preprint arXiv:2006.04768*, 2020.

[34] Z. Dai, H. Liu, Q. V. Le, and M. Tan, "CoAtNet: Marrying convolution and attention for all data sizes," in *Advances in Neural Information Processing Systems*, vol. 34, 2021, pp. 3965–3977.

[35] M. E. Sander, P. Ablin, and G. Peyré, "Do residual neural networks discretize neural ordinary differential equations?" *arXiv preprint arXiv:2205.14612*, 2022.

[36] C. Finlay, J.-H. Jacobsen, L. Nurbekyan, and A. M. Oberman, "How to train your neural ODE: The world of Jacobian and Kinetic regularization," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2020, pp. 3154–3164.

[37] J. Pan, A. Bulat, F. Tan, X. Zhu, L. Dudziak, H. Li, G. Tzimiropoulos, and B. Martinez, "EdgeViTs: Competing light-weight CNNs on mobile devices with Vision Transformers," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2022, pp. 294–311.

[38] H. Cho, D. Kim, S.-E. Hwang, and J. Park, "SpARC: Token similarity-aware sparse attention transformer accelerator via row-wise clustering," in *Proc. 61st ACM/IEEE Design Automation Conf. (DAC)*, 2024, pp. 181:1–181:6.

[39] J. Gao, W. Ji, F. Chang, S. Han, B. Wei, Z. Liu, and Y. Wang, "A systematic survey of general sparse matrix-matrix multiplication," *ACM Computing Surveys*, vol. 55, no. 12, pp. 1–36, 2023.

[40] L. Song, Y. Chi, A. Sohrabizadeh, Y.-K. Choi, J. Lau, and J. Cong, "Sextans: A streaming accelerator for general-purpose sparse-matrix dense-matrix multiplication," in *Proc. ACM/SIGDA Int. Symp. Field-Programmable Gate Arrays (FPGA)*, 2022, pp. 65–77.

[41] Y. Nagahara, J. Yan, K. Kawamura, M. Motomura, and T. V. Chu, "Sparse–sparse matrix multiplication accelerator on FPGA featuring distribute-merge product dataflow," in *Proc. Asia and South Pacific Design Automation Conf. (ASP-DAC)*, 2024, pp. 785–791.

[42] N. Srivastava, H. Jin, J. Liu, D. Albonesi, and Z. Zhang, "MatRaptor: A sparse–sparse matrix multiplication accelerator based on row-wise product," in *Proc. IEEE/ACM Int. Symp. Microarchitecture (MICRO)*, 2020, pp. 766–780.

[43] W. Ahrens and E. G. Boman, "On optimal partitioning for sparse matrices in variable block row format," *arXiv preprint arXiv:2005.12414*, 2020.

[44] M. Pligouroudis, R. A. G. Nuno, and T. J. Kazmierski, "Modified compressed sparse row format for accelerated FPGA-based sparse matrix multiplication," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, 2020, pp. 1–5.

[45] J. O. Maranga, J. J. Nnko, and S. Xiong, "Learned active contours via Transformer-Based deep convolutional neural network using canny edge detection algorithm," *Signal, Image and Video Processing*, vol. 19, no. 3, p. 222, 2025.

[46] M. Zhang, J. Cao, K. Shi, K. Zhao, G. Zhang, J. Yu, and K. Wang, "FNM-Trans: Efficient FPGA-based transformer architecture with full N:M sparsity," in *Proc. 61st ACM/IEEE Design Automation Conf. (DAC)*, 2024, pp. 191:1–191:6.

[47] S. Lu, M. Wang, S. Liang, J. Lin, and Z. Wang, "Hardware accelerator for multi-head attention and position-wise feed-forward in the Transformer," in *Proc. IEEE Int. System-on-Chip Conf. (SOCC)*, Sep. 2020, pp. 84–89.

[48] B. Li, S. Pandey, H. Fang, Y. Lyv, J. Li, J. Chen, M. Xie, L. Wan, H. Liu, and C. Ding, "FTRANS: Energy-efficient acceleration of transformers using FPGA," in *Proc. ACM/IEEE Int. Symp. Low Power Electronics and Design (ISLPED)*, 2020, pp. 175–180.

[49] H. Peng, S. Huang, T. Geng, A. Li, W. Jiang, H. Liu, S. Wang, and C. Ding, "Accelerating transformer-based deep learning models on

FPGAs using column balanced block pruning," in *Proc. Int. Symp. Quality Electronic Design (ISQED)*, Apr. 2021, pp. 142–148.

[50] C. Fang, A. Zhou, and Z. Wang, "An algorithm-hardware co-optimized framework for accelerating N:M sparse Transformers," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 30, no. 11, pp. 1573–1586, 2022.

Sichen Liu received the B.E. degree from Shandong University of Science and Technology, Qingdao, China, in 2024. He is currently pursuing a Ph.D. degree at the University of Sheffield, U.K. His research interests include Transformer acceleration and sparse network acceleration through hardware–software co-design.

Yuqin Zhao is currently a Research Associate in AI for verification and testing automation at the University of Edinburgh. His research focuses on integrating artificial intelligence into electronic design automation (EDA) workflows for design, verification, and testing, alongside broader interests in hardware-accelerated SoC design and next-generation EDA tool development. He received his Ph.D. in Electronic and Electrical Engineering from the University of Sheffield, UK, where his research centered on intelligent High-Level Synthesis (HLS) and AI-enhanced EDA methodologies for FPGA and ASIC design, with applications in software-defined radio (SDR) and AI-accelerated systems. He previously earned his M.Sc. from Nanyang Technological University (NTU), Singapore, and his B.Eng. from Tamkang University.

Haotian Liu received the MSc degree in Microelectronic System Design from the University of Southampton, United Kingdom. He is currently pursuing a Ph.D degree at the University of Sheffield. His research interests include high-performance computing, sparse network acceleration, and architectural support for parallel programming.

Haihang Xia received the B.Eng. degree in Integrated Circuit Design and Integrated System from Hefei University of Technology, Hefei, China, in 2022 and the M.Sc. degree in Electronic and Electrical Engineering from University of Glasgow, Glasgow, U.K., in 2023. He is currently pursuing the Ph.D. degree with the School of Electrical and Electronic Engineering, University of Sheffield, Sheffield, U.K. His research interests include Spiking Neural Networks, neuromorphic computing, deep learning, AI acceleration, and digital systems design.

Xinyi Wang received the M.Sc. degree in High Performance Computing from the University of Edinburgh, Edinburgh, U.K. She is currently pursuing the Ph.D. degree with the Department of Electronic and Electrical Engineering, University of Sheffield, Sheffield, U.K. Her research interests include spiking neural networks and hardware design.

John Goodenough received the B.Sc. degree in Engineering Science from the University of Durham, U.K., in 1988, and the Ph.D. degree in VLSI Architecture from the University of Sheffield, U.K., in 1991. He is currently a Professor with the School of Electrical and Electronic Engineering, the University of Sheffield, U.K. He is also an Independent Consultant with textrium.io, U.K., a Technology Advisor with Silicon Catalyst, U.S.A., and a member of the U.K. Government Semiconductor Advisory Panel. He previously held multiple senior positions at Arm, U.K. and U.S.A. (2000–2022), including Distinguished Engineer Architect for Secure SoC, Vice President of Engineering Systems, Vice President of Research Collaboration, and Director of Design Technology and Automation. He was also a board member of SRC Research Scholars Program, U.S.A. (2019-2022). Earlier, he was Director of Technology at Infinite Designs Ltd., U.K. (1995–2000), and a Research Associate and Lecturer at the University of Sheffield, U.K. (1991–1995).

Charith Abhayaratne (Member, IEEE) received the B.E. degree in electrical and electronic engineering from The University of Adelaide, Australia, in 1998, and the Ph.D. degree in electronic and electrical engineering from the University of Bath, U.K., in 2002. He was a recipient of the European Research Consortium for Informatics and Mathematics (ERCIM) Post-Doctoral Fellowship (2002-2004) to carry out research at the Centre of Mathematics and Computer Science (CWI), The Netherlands, and the National Research Institute for Computer Science and Control (INRIA), Sophia Antipolis, France. He is currently a Senior Lecturer with the School of Electrical and Electronic Engineering, The University of Sheffield, U.K. His research interests include machine learning, computer vision, visual content security, and multidimensional signal processing. He has published over 100 peer reviewed papers in leading journals, conferences and book editions. He has served as an associate editor for IEEE Transactions on Image Processing (2019–2024), IEEE Access (2020–2025) and Elsevier Journal of Information Security and Applications JISA (2019–2025).

Tiantai Deng received his PhD from Queen's University Belfast and BEng from Harbin Institution of Technology. His experience within Academia and Industry equipped him with the right knowledge for Front-End Chip Design and AI-EDA design.