Journal of Electronics and Informatics (ISSN: 2582-3825)
www.irojournals.com/iroei/

Resource-Efficient FPGA Implementation of a Self-Attention Core for Edge AI

Janaki Rani M.[1], Mohana Divya S.[2], Kavya D.[3], Maniyammai B.[4]

Department of Electronics and Communication Engineering, Dr. M.G.R. Educational and Research Institute, Chennai, India.

E-mail: [1]janakirani.ece@drmgrdu.ac.in, [2]mohanadivya.vlsi@gmail.com, [3]Kaviyad749@gmail.com, [4]bmaniyammai64@gmail.com

Abstract

The Modern AI relies on transformer-based architectures, they require considerable resource requirements along with substantial memory requirements for implementation in resource-limited deployments due to their inherent computational complexity. The use of Field Programmable Gate Arrays (FPGAs) to create an efficient hardware accelerator for self-attention within transformers will allow for efficient use of hardware resources while enabling the use of transformers as part of an edge AI system. This research describes the implementation of such a design using the Xilinx Artix-7 FPGA platform with the Nexys DDR development board and demonstrates the capability of self-attention within an FPGA-based hardware accelerator to produce results in real time through a pipelined multiply-accumulate (MAC) architecture and finite state machine (FSM) based control structures. In addition, the implementation of this hardware accelerator was able to achieve functional accuracy when compared to the Python Golden Model. The hardware accelerator produced low levels of both hardware utilization and power consumption while at the same time providing a efficient computational performance. The architecture provides a unique solution for balancing performance, resource utilization and energy efficiency. FPGA-based domain-specific accelerators have demonstrated their ability to bridge the computational gap presented by transformer self-attention mechanisms and would be an appropriate method for implementing real-time inference of edge AI systems.

DOI: https://doi.org/10.36548/jei.2026.2.005
Received: 02.03.2026, received in revised form: 03.04.2026, accepted: 22.04.2026, published: 02.05.2026

Keywords: FPGA Accelerator, Self-Attention, Transformer Models, Edge Computing, Fixed-Point Design, Pipelined MAC, BRAM-Based Architecture, Low-Latency Processing, Energy-Efficient Computing.

1. Introduction

The growing trend in artificial intelligence applications has created a need for better implementation of deep learning models in resource-limited settings. Edge intelligence is a promising research direction for enabling on-device inference with limited computation and energy resources [1], [5]. Within the realm of deep learning models, Transformer architectures have become prominent owing to their success in processing sequential and contextual information.

Transformer models rely on the self-attention mechanism, enabling them to learn long-distance relationships by selectively weighting the input elements. While self-attention has proven to be effective, it brings high computational costs with it, as many matrix multiplications must be calculated with complexity increasing as the square of the sequence length. This leads to higher latency and memory consumption, hindering the deployment of such models to edge devices.

In recent studies, hardware acceleration approaches have been investigated. FPGA-based approaches have shown potential with parallel processing and low power consumption [2], [7]. Innovative designs tailored for resource-optimised attention mechanisms and hardware-software synergies have come up with efficient designs with low power consumption. But many current designs still prioritize throughput over efficiency, resulting in increased resource and design complexity, which presents challenges for low-power edge applications [8], [9].

Moreover, compression techniques such as quantization and model pruning have also been explored in edge AI applications to reduce the computational load [1], [5]. Although these techniques enhance efficiency, incorporating them in hardware designs while maintaining accuracy can be difficult.

In this research, we propose a resource-efficient FPGA accelerator for the self-attention mechanism that is suitable for edge AI applications. The new design aims to achieve resource-efficient processing using low hardware resources. It employs 16-bit fixed-point (Q8.8)

arithmetic, a pipelined multiply-accumulate (MAC) unit and a memory design centered around the use of Block RAM (BRAM) to achieve low latency and power consumption.

The primary contributions of this work are: (i) a resource-efficient implementation of the self-attention block useful for edge AI applications, (ii) a design with pipelined architecture to support continuous data processing to achieve high throughput, and (iii) an efficient memory design to reduce data movement, improving system performance. This work shows that accelerator designs on FPGAs can successfully address the computational constraints of the self-attention operation while keeping resource consumption and power consumption minimal.

2. Literature Survey

The recent studies on edge intelligence have recognised the need for efficient deployment of deep learning models on edge devices. Ngo et al. [1] and Yuan et al. [5] explained that while deep neural networks have the highest accuracy, they are too complex and energy-consuming for the edge devices. Both of these papers highlight the opportunity to leverage efficient computation and storage to enable real-time inference.

Several approaches have been explored to tackle the challenges. FPGA accelerators have been explored for their parallel and energy efficient design. Zhang et al. [2] designed a resource-efficient attention accelerator for Vision Transformers, pointing out improvements in efficiency by using specific hardware designs. Similarly, Zhou et al. [7] proposed a hardware/software co-designed inference accelerator for transformers on edge FPGAs; they adopted system-level design optimizations to achieve better performance.

Similarly, Dong et al. [8] have explored scheduling approaches to accelerate transformers on FPGAs in order to improve the utilization and reduce the execution time. Hu et al. [9] have also reported co-optimization of convolutional and self-attention models; they found that a combination of the two models provides higher performance, but could lead to more complicated designs. However, the current state of the art in optimizing FPGA designs mostly involves performance optimizations (latency and throughput) at the cost of increased resource usage. Hence, they are not well-suited to resource-constrained and low-power edge devices. What's more, some approaches use complicated scheduling techniques or mixed precision techniques, which increase the complexity of the system.

The optimized models can be trained using quantization to other model optimisation techniques, but there has been a lot of work on these techniques. For instance, Sali et al. [4], [10] explored a few optimisation methods for DNN on FPGA, and concluded that lower-precision arithmetic can save resource usage and energy. However, these techniques pose challenges when applying to hardware without accuracy loss.

Also, approaches for getting large models on to edge devices have been discussed in Ferreira et al. [3] and Tao [6], which highlight the importance of data format, and hardware-aware design. These works indicate a trade-off between accuracy, efficiency and resource is important for edge AI.

As can be seen from the above discussion, there are either efforts devoted to high-performance design and design optimisation with effective techniques. But a light-weight and resource-efficient design space for self-attention remains on the edge FPGAs. This study seeks to solve this problem by designing an energy-efficient FPGA hardware accelerator for the self-attention mechanism. This will give us a lightweight design with low hardware resource utilisation, but high computational efficiency (Fixed-point operations, pipelined architecture and BRAM memory).

3. Proposed Methodology

Our work is a domain-specific system accelerated by the field programmable gate array (FPGA) that can compute the self-attention for edge artificial intelligence (AI) applications. The proposed methodology is designed to reduce the latency, energy and resource usage of the acceleration system via computation, memory access and dataflow optimizations.

A dataflow approach has been adopted for the system operation. The data input (Query (Q), Key (K) and Value (V) matrices) is firstly pre-processed and converted to 16-bit fixed-point (Q8.8) format in the host. Sequently the data is transferred to the FPGA board through a Universal Asynchronous Receiver/Transmitter (UART). The FPGA then transforms the data into parallel format and store it in Block RAM (BRAM).

Once data is loaded a finite state machine (FSM) starts computation. The FSM orchestrates the computation, and synchronises memory operations with the calculations. Six states are present during the execution of the matrix multiplication: IDLE, LOAD, COMPUTE, ACCUMULATE, STORE and DONE. In the LOAD state, the input matrices are loaded into

BRAM. In the COMPUTE and ACCUMULATE states, dot-products between Q and K^T are calculated with a pipelined Multiply-Accumulate (MAC) data path.

The MAC is pipelined to increase its throughput. A multiplication and an addition are executed each clock cycle allowing data to be processed without interruption after an initial delay. This makes the compute unit very efficient in its operation.

BRAMs are used for memory. All intermediate values are stored locally. Memory access time is reduced and memory bandwidth is improved. Data can also be read from BRAM in parallel to keep the MAC unit busy, without stalling the pipeline.

Fixed-point arithmetic (Q8.8) is employed. This approach eliminates the need for floating-point numbers and allows minimizing the number of DSP slices, power consumption, while still providing sufficient accuracy for attention calculation.

After calculating the attention scores, they are stored in BRAM and transmitted to the connected host system via UART. The host subsequently performs the post processing operations (softmax and multiply with Value (V) matrix).

The self-attention operation computed by the proposed accelerator is mathematically defined as:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V$$

Efficiency and effectiveness are the key design considerations. The architecture takes advantage of the pipelining, internal memory and low-precision number representation techniques to achieve high-speed processing with low hardware costs, and hence facilitates real-time AI inference in the edge environment.

This work proposes a domain-specific, pipelined hardware accelerator on the Nexys DDR XC7A100TCSG324-1 FPGA (Xilinx Artix-7, 100K logic cells, 4.9 Mb block RAM, 240 DSP slices). The architecture follows a Harvard-style design, decoupling control logic from the computational data path.

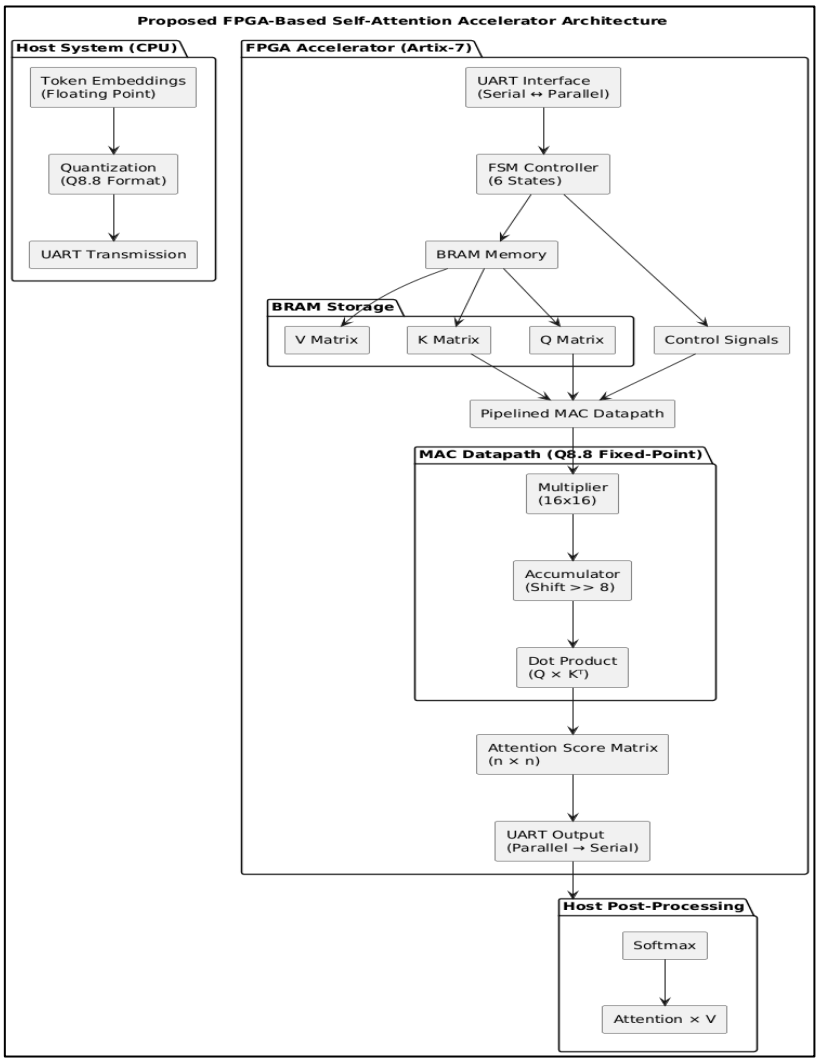

Figure 1. FPGA-Based Self-Attention Accelerator Architecture

Figure 1 illustrates the architecture of the proposed FPGA-based self-attention accelerator. The separation of control logic and datapath enables efficient execution and simplified control flow. The use of a pipelined MAC unit allows continuous data processing, while the BRAM-based memory structure ensures low-latency data access. This architectural design minimizes hardware resource utilization while maintaining high computational throughput, making it suitable for real-time edge AI applications.

The architecture of the proposed FPGA-based self-attention accelerator is shown in Figure 1, where the control and data path units are separated, in the Harvard-style design. The system is deployed in a heterogeneous environment, data preprocessing, post processing and the FPGA performing the attention computationally intensive computations. The input data is quantized and received by the UART interface and then translated into parallel format to be processed internally.

The control unit is built as a finite state machine (FSM), having six states, that performs the entire execution flow from reading data into memory to assisting with the computation sequence to writing results to memory. The Q matrix, K matrix, and V matrix are stored in dedicated on-chip BRAM arrays, providing a high-speed method of accessing data without any delays due to external memory access. The computation engine, which performs fixed-point multiplication and accumulation of the Q and K^T matrix dot products, is created as a MAC data-path with pipelined processing.

After calculating the attention scores, the control unit will pass the scores to the UART interface which will transmit back to the host system for continuing computations such as performing SoftMax normalizations and matrix multiplications with the V matrix. The system architecture is constructed such that it will use minimal FPGA resources, consume low power and execute at a high throughput, making this architecture suitable for use in real-time edge artificial intelligence applications.

3.1 System Overview and Host Interface

The system operates in a heterogeneous computing environment. The host pre-processes floating-point token embeddings by quantizing them into Q8.8 format before transmitting via UARTEach 16-bit word is sent as two 8-bit frames. The UART transfer time is:

$$T_{\text{UART}} = \frac{3nd_k \cdot 2 \cdot 10}{B}$$

Upon completion, the FPGA transmits the Attention Score matrix back to the host for post-processing and softmax application.

3.2 Fixed-Point Arithmetic — Q8.8 Format

The datapath replaces 32-bit IEEE 754 floating-point with 16-bit signed fixed-point arithmetic. A value x is quantized as $x\_fixed = round(x \times 2^8)$. After multiplication, the 32-bit product is rescaled: $p\_scaled = (a\_fixed \times b\_fixed) >> 8$. This reduces multiplier width from 32×32 to 16×16 bits, yielding approximately 4× fewer DSP slices and commensurate power reductions, while the 98.1 dB SQNR confirms negligible accuracy loss.

$$x_{\text{fixed}} = \text{round}(x \cdot 2^8)$$

3.3 Pipelined MAC Datapath

Each attention score element S_{ij} is the dot-product of row i of Q and column j of K^T. The two-stage pipelined MAC processes one multiply-accumulate per clock cycle after a two-cycle fill latency. At 100 MHz, this yields 100 million MACs/s sustained throughput.

Each attention score element is computed as:

$$T_{\text{FPGA}} = 2 + n^2 d_k$$

$$S_{ij} = \sum_{k=1}^{d_k} Q_{ik} \cdot K_{jk}$$

3.4 BRAM-Centric Memory Hierarchy

All Q, K, V matrices are stored in dedicated on-chip Block RAM partitions, completely eliminating off-chip DRAM accesses. The minimum BRAM requirement is $3 \cdot n \cdot d_k \cdot 16\ bits$. Single-cycle BRAM read latency ensures the MAC pipeline is never stalled. The memory bandwidth improvement ratio ranges from 10× to 20× at 100 MHz compared to DRAM access latencies of 100–200 ns.

3.5 Finite State Machine (FSM) Controller

The FSM orchestrates the accelerator through six deterministic states: IDLE, LOAD, COMPUTE, ACCUMULATE, STORE. The FSM manages memory addressing, datapath control signals, and status reporting back to the host CPU.

3.6 Hardware Platform

The proposed self-attention accelerator is implemented on the Nexys DDR development board featuring the Xilinx Artix-7 XC7A100T FPGA. The platform provides sufficient logic resources, DSP slices, and Block RAM to support the designed architecture. Additionally, the board includes on-chip peripherals such as UART interfaces, enabling efficient communication between the host system and the FPGA for data transfer and validation show in Figure 2.

Figure 2. Nexys DDR Development Board with Xilinx Artix-7 FPGA

The Figure 2 presents Nexys DDR FPGA development board used for implementation. The availability of sufficient logic resources, DSP slices and Block RAM enables efficient realization of the proposed accelerator. The integrated UART interface supports reliable data communication between the host system and FPGA, validating the practical feasibility of the design.

Figure 3. Place-and-Route Floorplan of the Synthesized Self-Attention Accelerator on the Artix-7

The Figure 3 shows the place and floor plan of the synthesized design using Vivado Design Suite, the reference number DSP48E1 is shown in the block arithmetic which nexys which has 100MHz of frequency for clock speed. Figure 3 shows the place-and-route floorplan of the synthesized design, highlighting the distribution of DSP blocks, logic elements, and BRAM resources across the FPGA fabric. The balanced placement with minimal routing congestion indicates efficient hardware mapping, which supports stable operation at 100 MHz.

This confirms that the proposed design achieves optimal resource utilization and reliable performance.

Figure 3 illustrates the place-and-route floorplan of the synthesized self-attention accelerator generated using the Vivado Design Suite. The floorplan shows the spatial distribution of key hardware resources, including DSP48E1 blocks used for arithmetic operations, logic elements (LUTs and registers), and Block RAM utilized for data storage. The design demonstrates efficient resource allocation with minimal congestion, ensuring balanced utilization across the FPGA fabric. The placement supports stable operation at the target clock frequency of 100 MHz while maintaining high throughput and low latency, validating the effectiveness of the proposed hardware architecture.

3.7 Proposed Self-Attention Computation Algorithm

The computation of the self-attention mechanism is implemented using a structured hardware-aware algorithm optimized for FPGA execution. The algorithm focuses on efficient matrix multiplication between the Query (Q) and Key (K^T) matrices using a pipelined MAC architecture.

Algorithm 1: FPGA-Based Self-Attention Score Computation

Input: Quantized matrices $Q \in \mathbb{R}^{n \times d_k}, K \in \mathbb{R}^{n \times d_k}$
Output: Attention score matrix $S \in \mathbb{R}^{n \times n}$

Initialize all elements:

$$S_{ij} = 0 \ \forall i, j$$

Load matrices Q and K into BRAM.
for $i = 1$ to n do
for $j = 1$ to n do
Initialize accumulator: $acc = 0$
for $k = 1$ to d_k do
Read Q_{ik} from BRAM
Read K_{jk} from BRAM
Multiply:

$$\text{temp} = Q_{ik} \cdot K_{jk}$$

Scale result:

$$temp = \frac{temp}{2^8}$$

Accumulate:

$$acc = acc + temp$$

end for

Store result:

$$S_{ij} = acc$$

end for
end for
Transmit matrix S to the host via UART.

The FPGA hardware has been mapped directly to the above described algorithm. The k loop (inner loop) will be accomplished through the use of a pipelined MAC unit; therefore, an individual multiply/accumulate will occur for every clock cycle after the pipeline has been set up. BRAM gives access to parallel data for continuous feeding of data to the computation unit and eliminating stalls.

The FSM coordinates the execution of the algorithm by managing memory reads, MAC operations, and data storage. This methodical approach will contribute to maximizing hardware resources, as well as achieve high throughput and low latency.

4. Implementation and Verification

4.1 HDL Design and Tool Flow

The accelerator was described in synthesizable Verilog HDL with three primary modules: (i) UART transceiver for host communication; (ii) FSM controller governing operation sequencing; and (iii) MAC datapath for pipelined arithmetic. Synthesis and place-and-route targeting the Nexys DDR XC7A100TCSG324-1 was executed using Xilinx Vivado. Block RAM inference was explicitly constrained for all memory arrays larger than 64×16 bits. The post-place-and-route timing report confirmed zero violations at 100 MHz.

4.2 Python Golden Model

A functionally equivalent Python Golden Model replicates the hardware's Q8.8 fixed-point arithmetic precisely, including truncation after each multiplication. The model accepted the same quantized inputs, computed QK^T dot products element-by-element, and exported reference results. Hardware simulation output was automatically diff-compared against this reference, with any bit-level discrepancy flagged as a verification failure.

4.3 Testbench and Simulation

A self-checking Verilog testbench instantiated the complete DUT and applied stimulus by: asserting system reset, driving UART with pre-encoded serial bit streams for Q, K, V matrices, monitoring the done flag, and comparing outputs byte-by-byte against the Golden Model using readmemh files. Multiple test vectors with varying matrix sizes, boundary conditions, and alternating sign patterns were applied to stress-test FSM transitions and accumulator overflow behavior.

5. Results and Discussion

Behavioral simulation in ModelSim yielded 100% bit-accurate match across all test cases against the Python Golden Model, confirming correct FSM sequencing, memory addressing, MAC accumulation without precision loss, and UART frame reconstruction without errors.

Physical synthesis confirmed the design fits comfortably within the Artix-7 device budget with exceptionally low utilization. The Table 1 shows the reports for Look up tables, registers, latches and muxes occupied by the design.

In the Table 1, the resource utilization such as DSP Blocks, Unique control sets are compared. The average utilization of Look up tables is 0.05 % of the total available LUTs in the board.

Table 1. Resource Utilization on Nexys DDR XC7A100TCSG324-1

Resource Type	Used	Available	Utilization (%)
Slice LUTs (Logic)	34	63,400	0.05%
Slice Registers	10	126,800	<0.01%
DSP48E1 Blocks	3	240	1.25%
Block RAM Tiles	6	135	4.44%
Unique Control Sets	3	15,850	0.02%

The extremely low LUT and register utilization (below 0.05%) is a direct consequence of the Harvard-style FSM-based control, which trades parallel spatial unrolling for sequential

pipelining, maximizing resource efficiency. Substantial headroom remains for future extensions such as Multi-Head Attention parallelism and on-chip softmax.

The Figure 4 shows the power analysis report which shows the following parameters:

On-chip power analysis demonstrates total power of 0.122 W, dominated by device static power (0.104 W, 85%). Dynamic power is only 0.018 W (15%), confirming the effectiveness of the BRAM-centric architecture. Reducing from 32-bit to 16-bit operand widths yields approximately 75% dynamic power savings.

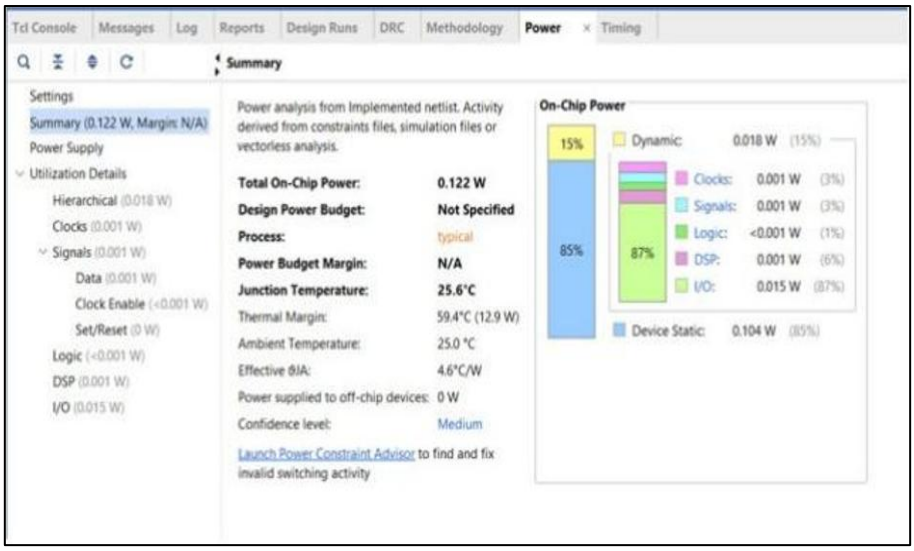

Figure 4. On-Chip Power Analysis Report

The FPGA accelerator achieves near-100% arithmetic unit utilization via the two-stage pipeline, single-cycle BRAM access eliminating the 10–20× DRAM latency penalty, and elimination of off-chip bus-switching activity. These architectural advantages collectively yield substantial performance speedup for the core QK^T matrix multiplication, directly translating to reduced inference latency for attention-based AI models at the edge.

6. Conclusion

This research work presents a resource-efficient FPGA-based implementation of the self-attention mechanism for edge AI applications. The proposed accelerator addresses the high computational and memory demands of self-attention by utilizing fixed-point arithmetic, a pipelined MAC Datapath, and a BRAM-centric memory architecture. The use of an FSM-based control structure enables efficient coordination of computation and data flow, resulting in

reduced latency and optimized hardware utilization. The implementation achieves full functional accuracy when validated against a Python Golden Model, while demonstrating extremely low resource utilization and low on-chip power consumption. These results confirm that the proposed design provides an effective balance between performance, resource efficiency, and energy consumption, making it suitable for real-time inference in resource-constrained environments. Overall, this work demonstrates that FPGA-based domain-specific accelerators offer a practical and scalable solution for accelerating self-attention mechanisms at the edge. Future work will focus on extending the design to support complete Transformer architectures, including multi-head attention, on-chip softmax computation, and scalability for larger and variable-length sequences.

References

[1] Ngo, Dat, Hyun-Cheol Park, and Bongsoon Kang. "Edge Intelligence: A Review of Deep Neural Network Inference in Resource-Limited Environments." Electronics 14, no. 12 (2025): 2495.

[2] Zhang, Wenbo, Yan Zhang, Yiqi Liu, Lingjie Wu, and Xingtong Hu. "REATA: An Efficient Vision Transformer Accelerator Featuring a Resource-Optimized Attention Design on Versal ACAP." ACM Transactions on Reconfigurable Technology and Systems 19, no. 1 (2026): 1-32.

[3] Ferreira, Lucas, Mariana Silva, Thiago Costa, Ana Beatriz Rocha, Rafael Almeida, Camila Oliveira, João Pedro Nunes, and Gulnaz Rati. "Bringing Foundation Models to the Edge with Efficient Deployment Strategies." Authorea Preprints 2025.

[4] Sali, Safa Mohammed, Mahmoud Meribout, and Ashiyana Abdul Majeed. "Real Time FPGA Based CNNs for Detection, Classification, and Tracking in Autonomous Systems: State of the Art Designs and Optimizations." 2025, arXiv preprint arXiv:2509.04153.

[5] Yuan, Haitao, Jing Bi, Ziqi Wang, Jia Zhang, MengChu Zhou, and Rajkumar Buyya. "Multi-Perspective and Energy-Efficient Deep Learning in Edge Computing." IEEE Internet of Things Journal 2025, vol. 13, no. 3, 3988-4003.

[6] Tao, Shuailin. "Edge-centric AI for Biomedical Signals: Efficient Representation and Intelligent Processing." PhD diss., Nanyang Technological University, 2025 10.32657/10356/204832.

[7] Zhou, Shuai, Sisi Meng, Huinan Tian, Jun Yu, and Kun Wang. "Edge-BiT: Software-Hardware Co-Design for Optimizing Binarized Transformer Networks Inference on Edge FPGA." In Proceedings of the 43rd IEEE/ACM International Conference on Computer-Aided Design, 2024, 1-9.

[8] Dong, Jiale, Wenqi Lou, Hao Wu, Zhendong Zheng, Yunji Qin, Lei Gong, Chao Wang, and Xuehai Zhou. "MoE-Sched: Enabling Efficient FPGA Deployment of Mixture-of-Experts Vision Transformers via Coordinated Scheduling." IEEE Transactions on Very Large Scale Integration (VLSI) Systems 2025, vol. 34, no. 1, 104-117.

[9] Hu, Wei, Heyuan Li, Fang Liu, and Zhiyv Zhong. "Hardware and Software Co-Optimization of Convolutional and Self-Attention Combined Model Based on FPGA." In Asia-Pacific Web (APWeb) and Web-Age Information Management (WAIM) Joint International Conference on Web and Big Data, Singapore: Springer Nature Singapore, 2023, 328-342.

[10] Sali, Safa Mohammed, Mahmoud Meribout, and Ashiyana Abdul Majeed. "Real Time FPGA Based Transformers & VLMs for Vision Tasks: SOTA Designs and Optimizations." 2025, arXiv preprint arXiv:2509.04162.