Edge QLoRA: Hardware Accelerated QLoRA for On-Device LLM Fine Tuning

Goonetilleke P, Jayakody J A K, Warushavithana N D
Silicon Edge, Submission Number 6481
University of Moratuwa

April 12, 2025

Contents

1 Introduction **4**

2 Background Research **5**
2.1 QLoRA and Efficient Fine Tuning . . . 5
2.2 Challenges in Implementing Large Language Models in Edge Devices . . . 5
2.3 FPGA for LLMs . . . 6

3 Goal and Objectives **6**
3.1 Goal . . . 6
3.2 Objectives . . . 6

4 Design Process **6**
4.1 Problem Statement . . . 6
4.2 Functional Specification . . . 7
4.3 Proposed Design . . . 7
4.3.1 Design Components . . . 7
4.3.2 Parallel Pipelines for Inference and Fine-Tuning . . . 8
4.4 Analysis of Final Design . . . 9
4.4.1 Control Plane . . . 9
4.4.2 Data Plane . . . 9
4.5 Testing and Implementation of Final Design . . . 9

5 Results and Discussion **11**
5.1 Quantitative Results . . . 11
5.2 Qualitative Insights . . . 11
5.3 Case Study . . . 11

6 Conclusion **12**

7 References **13**

List of Tables

1 Performance Metrics . . . 10

List of Figures

1 Block Diagram . . . 7
2 Functional Block Diagram . . . 9

Abstract

Deploying and fine-tuning large language models (LLMs) on edge devices is challenging due to their high computational demands, memory requirements, and power consumption. While Low-Rank Adaptation (LoRA) reduces fine-tuning complexity, its memory footprint remains prohibitive for resource-constrained devices. Quantized LoRA (QLoRA) addresses this by combining 4-bit quantization with LoRA, significantly reducing memory usage while preserving model performance. However, efficient hardware acceleration is still needed to meet the real-time processing and energy constraints of edge applications. This work presents Edge QLoRA, an FPGA-accelerated implementation of QLoRA designed for on-device LLM fine-tuning. We propose a custom hardware architecture leveraging FPGA reconfigurability to optimize quantization, dequantization, and matrix operations—key components of QLoRA. Our design integrates systolic arrays for parallelized matrix multiplication, on-chip BRAM caching for efficient memory management, and power-optimized pipelines for inference and fine-tuning. Key contributions include: A memory-efficient FPGA implementation of QLoRA, reducing TinyLlama's (1.1B parameters) footprint to ¡300MB while enabling fine-tuning on edge devices, hardware-accelerated 4-bit quantization/dequantization with custom IP cores, improving computational throughput, and benchmarks demonstrating superior latency, power efficiency, and memory savings compared to GPU-based QLoRA, validated through simulations targeting a Vega processor-based FPGA. Our preliminary estimates (python based software implementations) indicate a 99% speedup in processing time (for quantiztion and dequantization) and we expect 75% reduction in model size compared to FP16 baselines, making LLM fine-tuning feasible on edge hardware. This approach bridges the gap between cloud-based LLMs and edge deployment, enabling privacy-preserving, low-latency applications in healthcare, IoT, and robotics. Future work will focus on expanding support for larger models and dynamic quantization strategies.

1 Introduction

Deploying large language models (LLMs) on edge or embedded devices is extremely challenging. These models demand substantial memory to store parameters and extensive computational power. This computational power is typically provided by GPUs, which are power hungry and expensive, removing them as a viable option for edge/embedded applications.

Furthermore, fine-tuning LLMs on edge devices is nearly impossible due to their resource constraints. While cloud-processing could address these issues, it adds latency and poses a privacy concern.

Recent advancements such as Low-Rank Adaptation (LoRA) provide a way to accelerate fine-tuning while reducing computational requirements. However, LoRA alone does not address the memory demands of LLMs. This limitation is tackled by Quantized LoRA (QLoRA), which applies quantization techniques to compress model parameters, drastically reducing memory usage and improving computational efficiency. By quantizing model weights to lower bit-width representations, QLoRA reduces memory footprints and accelerates matrix multiplication, the core operation in LLMs. This makes LLM deployment on edge devices more feasible. We aim to implement QLoRA on FPGA-based systems, leveraging their high computational throughput, low power consumption, and reconfigurability. Specifically, we will:

- Design custom IP cores for quantization and matrix multiplication, optimized for QLoRA.
- Minimize resource usage to fit the constraints of embedded systems.
- Evaluate performance gains in speed, memory efficiency, and power consumption compared to traditional GPU-based solutions.

Implementing QLoRA on FPGAs would enable efficient, low-latency deployment of LLMs on resource-constrained devices. This approach offers a promising alternative to cloud processing, enhancing privacy, robustness, and efficiency for various applications.

2 Background Research

2.1 QLoRA and Efficient Fine Tuning

Deploying large language models (LLMs) on edge devices is challenging due to their high memory requirements, computational complexity, and power consumption. Edge devices, such as embedded systems, typically lack the processing power and memory capacity needed for LLM inference and fine-tuning. GPUs, which are commonly used for LLM deployment, are power-hungry and expensive, making them impractical for most edge applications. Cloud processing offers a partial solution, but it introduces latency, dependency on external networks, and potential privacy concerns when transmitting sensitive data. Efficiently deploying and fine-tuning LLMs on edge devices could enhance privacy, reduce latency, and improve power efficiency, making it a valuable objective for various real-time, resource-constrained applications.

LoRA: Low Rank Adaptation for large language models (LLM) is a method to fine tune a large language model by only training two low rank matrices, task specific LoRA modules, while the pretrained model weights are freezed. This method drastically reduces the inference latency and input sequence length while retaining the high model quality. [1]. LoRA can be used in fine tuning LLM on edge devices but the main constraint is the inefficiency that is caused by the high memory usage of these large models.

QLoRA is a novel method for efficiently finetuning large language models by 4-bit quantization and using Low Rank Adapters (LoRA). This approach optimizes memory usage without affecting performance. QLoRA enables multi billion parameter models to be finetuned on a single GPU, while preserving the original 16-bit precision of the model weights. QLoRA incorporates three main features to achieve this performance: 4-bit normal float (NF4) data type which is optimal for the normally distributed model weights; double quantization to further reduce memory usage by quantizing the quantization constants, and paged optimizers to handle memory spikes [2].

2.2 Challenges in Implementing Large Language Models in Edge Devices

Deploying large language models (LLMs) on edge devices presents several critical challenges due to their computational demands and resource constraints. Edge devices, such as embedded systems, often lack the necessary processing power, memory LLM inference and fine tuning. Their low-power CPUs struggle with the extensive computational complexity, while memory constraints hinder the storage and execution of large-scale models [3]. Additionally, power efficiency is a significant concern, as these devices typically operate on battery power or limited energy sources. The high energy consumption of LLMs, caused by continuous matrix multiplications and memory access, leads to excessive power draw and thermal management issues that can compromise reliability .

[4] Connectivity limitations such as reliance on cloud-based processing introduces latency, dependency on external networks, and potential security risks when transmitting sensitive data . Furthermore, latency and real-time processing requirements pose additional challenges for edge applications like autonomous systems. Another main issue in implementing LLMs in on edge devices is the inefficiency of on device training/ fine tuning which is normally done on cloud rather than on device. edge [5].

Mitigating These Challenges with QLoRa on FPGA To address these challenges, we propose a QLoRa approach implemented in FPGA based on the Vega processor core enabling efficient inferencing and fine-tuning of LLMs on edge devices while optimizing power consumption and computational efficiency. By leveraging FPGA-based acceleration, QLoRa can significantly reduce latency in inference and facilitate fine tuning while reducing the model size, making LLM deployment feasible even in constrained edge environments.

2.3 FPGA for LLMs

FPGAs can address the above limitations through their customizability. Unlike the fixed architectures of CPUs and GPUs, FPGAs can be configured to create hardware that is tailored to meet the specific computational and memory demands of LLMs on embedded/edge devices.

A few examples include:

- Parallizing computations, which would speed up matrix multiplications, one of the most computationally intensive operations in LLMs[6]
- Optimization of memory and power utilization [6]

Which would address concerns such as:

- Performance (by eliminating the need for CPU based inference)
- Data-Privacy (by eliminating the need for cloud-based processing)
- Resource limitations on embedded/edge devices by reducing utilization

Further, FPGA-based LLM accelerators show promising results:

- FTRANS (2020) achieved 16x model size reduction and 27x CPU performance gains [7]
- MHA(2020) reported 14.6x speedups [8]
- Tzanos(2022) achieved 2.3x BERT speedup [9]
- NPE (2021) reduced power consumption by 4-6x [10]

3 Goal and Objectives

3.1 Goal

Design and develop hardware accelerator on FPGA for QLoRA fine-tuning of LLMs on edge devices.

3.2 Objectives

- Efficient integration of the accelerator with the VEGA processor.
- Achieve limited memory usage ($<$ 300MB) for 1.1 B parameter models.
- Demonstrate real-world applicability of the accelerator.
- Accuracy/Memory efficiency/power efficiency/speedup

4 Design Process

4.1 Problem Statement

Deploying LLMs on edge devices is challenging due to their high demand in computation, memory, and power. Traditional methods rely on cloud-based processing which adds latency and security risks. Although Low Rank Adaptation (LoRA) reduces the fine-tuning computational complexity, its high memory requirement remains a barrier for edge deployment [11]. This is addressed by Quantized LoRA (QLoRA), which improves the memory efficiency via NF4 quantization, double quantization, and paged optimizes. However, hardware acceleration is still needed to meet realtime processing and energy constraints of edge devices.

4.2 Functional Specification

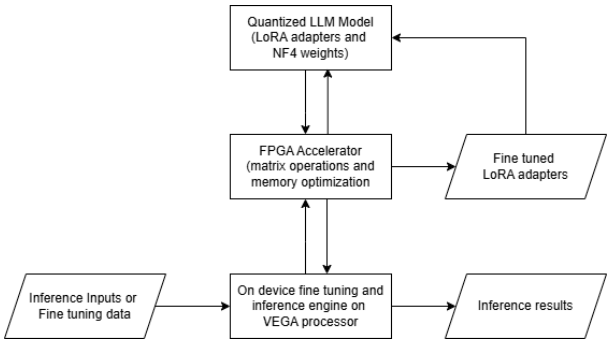

Figure 1: Block Diagram

Functional flow of the implementations is depicted in figure [2] and the functional requirements of the blocks are as follows. Quantized LLM model requires efficient quantization and dequantization of 4 bit normal floats, double quantization for weights and quantization constants, and low rank matrices integration for LoRA adapters. Moreover, the FPGA accelerator requires parallelized matrix operations with systolic arrays and tiling, efficient memory management with on chip BRAM chaching, and paged optimizing. The inference and fine tuning engine demands low latency inference and on device fine tuning. Additionally, power and resource management should be implemented efficiently in the FPGA to meet the device constraints.

4.3 Proposed Design

This section presents the proposed design leveraging FPGA capabilities, specifically targeting efficient matrix operations and model fine-tuning with Low Rank Adapters (LoRA). The design focuses on utilizing Intellectual Property (IP) cores tailored for quantization, double quantization, dequantization, and double dequantization to optimize both inference and fine-tuning processes. Additionally, dedicated IP cores for matrix multiplications will be implemented to accelerate computational tasks, enhancing overall performance.

4.3.1 Design Components

IP for Quantization and Dequantization

- Implement IP cores for quantization and dequantization processes. This includes support for double quantization and dequantization to handle complex data transformations efficiently.
- Separate IP cores will be designed for quantization and dequantization operations, enabling parallel processing during both inference and fine-tuning.

IP for Matrix Multiplications

- Develop IP cores specialized in matrix operations to leverage FPGA resources effectively. This includes utilizing FPGA strengths such as DSP slices for fast arithmetic operations and block RAM for efficient data storage and retrieval.

- Implement systolic arrays for matrix multiplication, using a **Weight Stationary (WS)** data flow to maintain static weights during fine-tuning.

Implementation Strategy

- **Utilization of FPGA Resources:**
  - Leveraging FPGA capabilities such as 50,950 logic slices with 6-input LUTs and 8 flip-flops, 16 Mbits of fast block RAM, and 840 DSP slices will optimize matrix multiplication tasks.
  - The design will exploit FPGA's clock management tiles and high internal clock speeds exceeding 450MHz to ensure rapid computation and synchronization.
- **Systolic Array Implementation:**
  - Implementing systolic arrays will enhance parallelism and optimize matrix multiplication efficiency. This approach utilizes specialized data flow strategies like Weight Stationary (WS), ideal for maintaining static weights during fine-tuning.
  - Each Processing Element (PE) within the systolic array will be designed to accept data from neighboring PEs, perform Multiply-Accumulate (MAC) operations, and forward results to the next PE.

4.3.2 Parallel Pipelines for Inference and Fine-Tuning

To maximize throughput and efficiency, **separate parallel pipelines** will be implemented for inference and fine-tuning processes. This separation allows independent optimization of each path, minimizing latency and avoiding contention for resources.

Inference Pipeline

- Primarily involves matrix multiplication using pre-trained weights stored in FPGA block RAM.
- Operations include **dequantization of inputs**, matrix multiplication using systolic arrays, and **quantization of results**.
- Parallelism is achieved by distributing matrix multiplication tasks across multiple DSP slices and block RAM units.
- The inference pipeline maintains a steady data flow by implementing double buffering, allowing computation and memory access to occur simultaneously.

Fine-Tuning Pipeline

- Involves matrix multiplication using pre-trained weights and **Low Rank Adapters (LoRA)** for model adaptation.
- Operations include **dequantization of weights and inputs**, matrix multiplication using modified weights, and **quantization of updated weights** before storage.
- Additional steps for fine-tuning involve quantizing updated weights and LoRA matrices before storing them back in block RAM or external memory.
- Uses **Weight Stationary (WS)** data flow strategy, where weights remain static, allowing efficient reuse of pre-trained model parameters.

Concurrency Management

- To avoid bottlenecks, separate pipelines for inference and fine-tuning operate concurrently, with dedicated resources allocated for each process.
- Quantization and dequantization units are designed to **operate concurrently with matrix multiplication**, ensuring that data preparation does not hinder computational throughput.
- Pipelining ensures that while one stage of computation is being processed, the next stage is already being prepared, enhancing overall efficiency.

4.4 Analysis of Final Design

4.4.1 Control Plane

Vega processor handles the control flow: manages model loading, task scheduling, and FPGA integration, pre/post-processing of tokens, and fine tuning/inference logic.

4.4.2 Data Plane

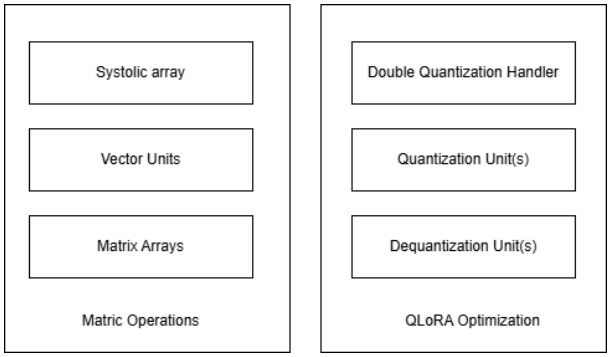

Figure 2: Functional Block Diagram

2. Detailed Dataflow
2.1 Inference Mode
Input Token → Vega CPU (preprocessing).
FPGA Loads Quantized Weights (4-bit NF4) from DDR, dequantizes to FP16 them and stores in BRAM for computation
Systolic Array computes: Output = (Wquant × X) + (A × B) × X (LoRA adapter fusion).
Results streamed back to Vega CPU for post-processing.
2.2 Fine-Tuning Mode Forward Pass: Same as inference.
Backward Pass: Gradient computed via systolic array.
Only LoRA matrices (A B) updated (frozen base weights).
Paged Optimizers: FPGA manages memory spikes via on-chip buffering.
Quantize and store fine-tuned weights in DDR3 RAM.

4.5 Testing and Implementation of Final Design

Testing and implementation involve evaluating the FPGA-based QLoRA accelerator to ensure efficient fine-tuning and inference of LLMs on edge devices. Initial tests will be conducted through software simulations to compare performance metrics like speed, memory usage, and accuracy against traditional approaches. After successful simulations, the design will be implemented on FPGA hardware, integrated

with the VEGA processor. Performance benchmarks, including latency, throughput, and power consumption, will be measured and compared with expected results. Adjustments and optimizations will be made as necessary to achieve the desired performance, ensuring the accelerator meets the project's objectives.

Metric	**Target**
Model Size Reduction	≥4x vs. full-precision model
Inference Latency	<100ms per token (edge-optimized)
Fine-Tuning Memory Usage	<300MB (for TinyLlama)
Power Consumption	<5W during peak inference
FPGA Utilization	<40% LUT/BRAM usage by proposed IPs

Table 1: Performance Metrics

5 Results and Discussion

5.1 Quantitative Results

Based on preliminary software simulations, we observed a 99% speedup in processing time for LLMs when applying quantization and dequantization techniques. These results were obtained by simulating the effects of these optimizations using Python-based models. We expect a similar speedup when the solution is implemented on FPGA hardware.

In terms of memory usage, we expect quantization (along with pruning) to lead to a reduction of model size by up to 75%, significantly improving the feasibility of deploying these models on edge devices.

5.2 Qualitative Insights

The FPGA-based accelerator offers significant advantages over traditional CPU or GPU-based approaches in terms of power efficiency and real-time processing capabilities. By reducing memory requirements and accelerating computation, it facilitates faster inference and fine-tuning on edge devices, which would be otherwise impossible due to hardware limitations. However, the main limitation remains in the complexity of implementing a highly efficient, fully optimized FPGA solution for all stages of the quantization and dequantization processes, matrix multiplications, as well as pipelined stages. The hardware resource constraints and design time required to ensure minimal power consumption while maintaining high performance also present challenges.

5.3 Case Study

Edge-based Large Language Models (LLMs) are ideal for real-time, privacy-sensitive applications like healthcare chatbots. These systems need quick responses to assist patients and healthcare professionals, and cloud-based processing is not suitable due to network delays. FPGA-accelerated LLMs using QLoRA for fine-tuning and memory efficiency can process data locally, reducing latency and improving response times. In healthcare, protecting sensitive patient information is crucial. Storing and processing data in the cloud can pose security risks. By running LLMs on edge devices with FPGA accelerators, sensitive data stays on the device, ensuring privacy. These edge devices can also be fine-tuned to handle specific medical information without needing to send data to the cloud. With FPGA-based QLoRA, edge devices can run efficient, fine-tuned models while addressing key issues like latency, fine-tuning, and privacy. This approach enhances performance, security, and adaptability in healthcare chatbots, making it a practical solution for real-world applications where quick, secure, and context-aware AI is needed.

6 Conclusion

Deploying Large Language Models (LLMs) on edge devices has traditionally been problematic due high computational demands, memory constraints, and power consumption. Our approach—implementing Quantized LoRA (QLoRA) on FPGA-based systems—addresses these challenges by employing hardware acceleration, quantization techniques, and optimized matrix operations.

Our proposed FPGA-based accelerator significantly enhances the efficiency of LLM inference and fine-tuning by reducing memory usage, power consumption, and processing latency. The implementation of systolic arrays, parallel processing pipelines, and dedicated IP cores for quantization and matrix multiplication ensures optimal performance within the constraints of embedded systems.

Preliminary simulations indicate substantial speedups in processing time, with model size reductions of up to 75%, making real-time deployment on edge devices feasible. Additionally, our approach improves privacy by enabling on-device fine-tuning, reducing reliance on cloud-based processing. A case study in healthcare application highlights the practical benefits of this method, demonstrating how FPGA-accelerated LLMs can offer secure, low-latency AI solutions in critical applications.

Our work will focus on further optimizing the system design, estimations of the efficient model sizes and the IP block specifications, integrating the newly introduced IP blocks to seamlessly integrate with the VEGA processor to implement the optimized system of QLoRA-based LLM deployment in resource-constrained environments. With continued advancements, this approach has the potential to revolutionize real-time AI applications at the edge, unlocking new possibilities for intelligent, privacy-conscious, and efficient systems.

7 References

[1] E. J. Hu, Y. Shen, P. Wallis, *et al.*, *Lora: Low-rank adaptation of large language models*, 2021. arXiv: 2106.09685 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2106.09685.

[2] T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer, *Qlora: Efficient finetuning of quantized llms*, 2023. arXiv: 2305.14314 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2305.14314.

[3] Y. Zheng, Y. Chen, B. Qian, X. Shi, Y. Shu, and J. Chen, *A review on edge large language models: Design, execution, and applications*, Sep. 2024. DOI: 10.48550/arXiv.2410.11845.

[4] M. M. H. Shuvo, S. K. Islam, J. Cheng, and B. I. Morshed, "Efficient acceleration of deep learning inference on resource-constrained edge devices: A review," *Proceedings of the IEEE*, vol. 111, no. 1, pp. 42–91, 2023. DOI: 10.1109/JPROC.2022.3226481.

[5] M. M. H. Shuvo, S. K. Islam, J. Cheng, and B. I. Morshed, "Efficient acceleration of deep learning inference on resource-constrained edge devices: A review," *Proceedings of the IEEE*, vol. 111, no. 1, pp. 42–91, 2023. DOI: 10.1109/JPROC.2022.3226481.

[6] C. Kachris, "A survey on hardware accelerators for large language models," *Applied Sciences*, vol. 15, no. 2, p. 586, Jan. 2025, ISSN: 2076-3417. DOI: 10.3390/app15020586. [Online]. Available: http://dx.doi.org/10.3390/app15020586.

[7] B. Li, S. Pandey, H. Fang, *et al.*, "Ftrans: Energy-efficient acceleration of transformers using fpga," in *Proceedings of the ACM/IEEE International Symposium on Low Power Electronics and Design*, ser. ISLPED '20, Boston, Massachusetts: Association for Computing Machinery, 2020, pp. 175–180, ISBN: 9781450370530. DOI: 10.1145/3370748.3406567. [Online]. Available: https://doi.org/10.1145/3370748.3406567.

[8] 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 *2020 IEEE 33rd International System-on-Chip Conference (SOCC)*, 2020, pp. 84–89. DOI: 10.1109/SOCC49529.2020.9524802.

[9] G. Tzanos, C. Kachris, and D. Soudris, "Hardware acceleration of transformer networks using fpgas," in *2022 Panhellenic Conference on Electronics Telecommunications (PACET)*, 2022, pp. 1–5. DOI: 10.1109/PACET56979.2022.9976354.

[10] H. Khan, A. Khan, Z. Khan, L. B. Huang, K. Wang, and L. He, "Npe: An fpga-based overlay processor for natural language processing," in *The 2021 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, ser. FPGA '21, Virtual Event, USA: Association for Computing Machinery, 2021, p. 227, ISBN: 9781450382182. DOI: 10.1145/3431920.3439477. [Online]. Available: https://doi.org/10.1145/3431920.3439477.

[11] W. Wang, Y. Zhang, Z. Zhang, *et al.*, *Roma: A read-only-memory-based accelerator for qlora-based on-device llm*, 2025. arXiv: 2503.12988 [cs.AR]. [Online]. Available: https://arxiv.org/abs/2503.12988.