# Computationally Efficient FPGA-based Large Language Model Inference for Real-Time Decision-Making in Robotic Systems

Huaizhi Zhang[1], Tamim M. Al-Hasan[1], Xuqi Zhu[1], Jiacheng Zhu[1], Weiyong Si[1], Klaus D. McDonald-Maier[1], and Xiaojun Zhai[1]

***Abstract*— Integrating Large Language Models (LLMs) into modern robotic systems presents significant computational and energy constraint challenges, particularly for human-centered robotic applications. This paper presents a novel hardware optimization technique for deploying LLMs on resource-constrained embedded devices, achieving an up to 77% reduction in computational latency through an FPGA implementation in comparison to other popular embedded computing devices (e.g., CPU and GPUs). Additionally, we demonstrate our methodology by deploying a LLaMA 2-7B model on a Unitree Go2 robotic dog integrated with the proposed FPGA platform. The proposed optimization framework preserves real-time interaction capabilities while significantly reducing computational and energy overhead, facilitating efficient natural language processing for human-robot interaction in safety-critical and dynamic environments. Experimental results demonstrate that the FPGA-based LLaMA 2-7B implementation achieves up to 6.06-fold and 1.95-fold higher throughput compared to baseline CPU and GPU implementations while maintaining comparable inference accuracy. Furthermore, the proposed FPGA design surpasses existing state-of-the-art FPGA implementations, delivering a 30% improvement in computational efficiency.**

## I. Introduction

Recent years have witnessed an explosion in the capabilities of Large Language Models (LLMs), with GPT [1], BERT [2], and LLaMA [3] demonstrating exceptional performance across a broad array of Natural Language Processing (NLP) tasks. Advancements in transformer architectures [4], [5] have significantly improved both the efficiency and efficacy of language understanding and generation, as exemplified by BERT's bidirectional training for enhanced contextual comprehension [6] and GPT's coherent, contextually relevant text generation [7]. In parallel, the field of robotics has seen rapid progress in manipulation, navigation, and interaction capabilities, driven by machine learning breakthroughs [8]. As robots are deployed in diverse settings, from industrial facilities to healthcare, there is a growing need for natural language interfaces to translate complex commands and enable conversational interaction [9].

Integrating LLMs into robotic platforms promises to greatly enhance Human-Robot Interaction (HRI) [10]. Rather than relying on hand-held controllers or pre-coded scripts,

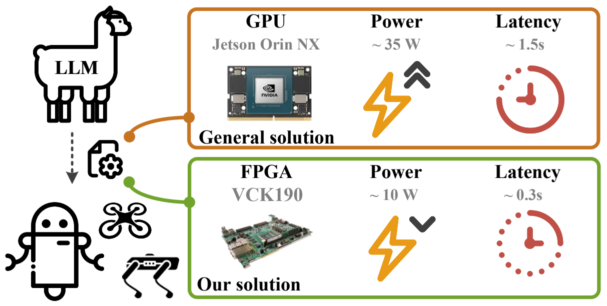

Fig. 1. Motivation of this research work: FPGA-enhanced HRI aiming to reduce latency and energy consumption in onboard computing, thereby improving real-time performance and efficiency.

users can delegate tasks through natural language and receive contextual explanations or feedback [11], [12]. This capability is particularly impactful in assistive and accessibility-focused scenarios, benefiting the elderly, visually impaired individuals, or users with limited mobility through a voice-driven interface [13]. By accurately interpreting language instructions and providing real-time guidance, a robot can provide transformative assistance in everyday life [14].

However, as shown in Fig. 1, the deployment of LLMs on physical robots faces the challenge of computational and energy overhead for current robotic systems. Large-scale transformer-based architectures require substantial processing power and memory bandwidth [15], which can be prohibitive on mobile robotics platforms where power and compute resources are limited. While cloud-based inference offloads heavy computations, it introduces latency, connectivity, and privacy issues. Consequently, onboard inference is highly desirable yet difficult to achieve efficiently under strict resource budgets. Researchers have begun exploring hardware acceleration and model compression techniques, such as pruning, quantization, and knowledge distillation, to reduce LLM computational requirements without sacrificing performance [16]. FPGAs further facilitate application-specific parallelism and customizable data pipelines [17], showing promise for lowering latency and energy consumption when integrated into real-time control loops [18].

In this paper, we propose a hardware-software co-optimization framework for efficient, on-board LLM inference on an AMD® Versal™ VCK190 FPGA platform. Our primary contributions are two-fold:

- **FPGA Acceleration of Transformer-based LLMs:** We detail a design flow that implements quantized, pruned, and pipeline-optimized variants of the LLaMA 2-7B implementation on an FPGA, achieving up to 6.06× and

*This work was supported by the UK Engineering and Physical Sciences Research Council through grants, EP/X015955/1, EP/X019160/1, EP/V000462/1, and EP/V034111/1. For the purpose of open access, the author has applied a Creative Commons Attribution (CC BY) license to any Author Accepted Manuscript version arising.

[1]All authors are with the School of Computer Science and Electronic Engineering, University of Essex, Wivenhoe Park, Colchester CO4 3SQ, United Kingdom {hz24245, tamim.almulla, xz18173, jz23222, w.si, kdm, xzhai}@essex.ac.uk

TABLE I

Overview of Selected Compute-Efficient LLM Approaches.

| Study | Model(s) | Platform(s) | Inference Speed (TPS) | Power / Efficiency | Performance Gains |
|---|---|---|---|---|---|
| **SpQR** [26] | LLaMA<br>(7B, 13B, 30B, 65B) | GPU (NVIDIA® A100) | 7B: 57.0 ± 2.4; 30B: 22.0 ± 0.9<br>13B: 44.0 ± 0.5; 65B: 12.0 ± 0.6 | N/A | Near-lossless compression;<br>up to 4× memory reduction |
| **GPTQ** [27] | GPT, OPT, BLOOM,<br>LLaMA (up to 175B) | GPU (NVIDIA® A100),<br>GPU (NVIDIA® A6000) | A100: 14.08<br>A6000: 7.69 | N/A | 2–4 bit post-training quant.<br>Minimal accuracy drop |
| **Additive Quant.** [28] | LLaMA 2<br>(7B, 13B, 70B) | CPU (Intel® Core™ i9-13900K),<br>GPU (NVIDIA® RTX™ 3090) | **GPU:** 7B: 65.3, 13B: 34.1, 70B: 6.7<br>**CPU**: 7B: 6.961, 13B: 4.180, 70B: 0.966 | N/A | 2–3 bit “extreme” compression;<br>Up to 8× memory reduction |
| **FlightLLM** [29] | LLaMA 2–7B | FPGA (Xilinx® Alveo™ U280),<br>vs. GPU (NVIDIA® Tesla® V100S) | FPGA: 55<br>(versus V100S: 45) | ~45 W<br>1.8× better cost efficiency<br>6.0× higher throughput/power | 1.2× throughput vs. V100S;<br>500× reduction in<br>compilation overhead |
| **HLSTransform** [30] | LLaMA 2<br>(up to 110M) | FPGA (Xilinx® Virtex™ UltraScale+ VU9P),<br>GPU (NVIDIA® RTX™ 3090),<br>CPU (Intel® Xeon™ E5-2686v4) | **256 tokens:** CPU: 23.21, GPU: 107.00, FPGA: 57.11<br>**1024 tokens:** CPU: 19.63, GPU: 107.24, FPGA: 57.11 | **Power (W)** / **Efficiency (mWh/token)**<br>CPU: 42.5 / 0.51 to 0.60<br>GPU: 126.9 to 130.6 / 0.33 to 0.34<br>FPGA: 9 / 0.04 | 2.46× speedup vs. CPU,<br>0.53× GPU speed,<br>12.75× less energy vs. CPU,<br>8.25× vs. GPU |
| **LLaMAF** [31] | TinyLLaMA<br>(1.1B) | FPGA (Xilinx® ZCU102) | step=64: 1.478<br>step=128: 1.424<br>step=256: 1.328 | Efficiency: 0.291 TPS/W<br>Baseline: 0.048 TPS/W | ~14–15.8× speedup<br>vs. PS-only,<br>6.1× better power efficiency |
| **Hasan** [32] | Various (up to 1B) | N/A | FP32: 50<br>INT8: 120<br>INT4: 150 | FP32: 10 W;<br>INT8: 6 W,<br>INT4: 4 W<br>~60% power reduction | QAT vs. PTQ analysis;<br>up to 68% model size<br>reduction |

1.95× throughput improvements compared to baseline CPU/GPU implementations while retaining similar inference accuracy with the original model.

- **End-to-End Real-Time Robot Control Pipeline:** Through tight integration with the Unitree Go platform, we demonstrate low-latency language-based command processing and control-loop decision-making, which is crucial for tasks involving human-robot dialogue and responsive navigation.

These enhancements address the central challenge of real-time performance while enabling advanced assistive HRI. Specifically, we illustrate how an embedded LLM on a quadruped can verbally interact with individuals, parse natural language requests, and deliver situational guidance in real-time, a crucial capability for high-stakes environments such as healthcare facilities, eldercare homes, and public-access buildings. Despite clear benefits, the complexity of LLM-based systems requires careful model compression and accelerator design to meet tight performance budgets in legged robotics. Balancing these optimizations with functional requirements, such as conversational fluency and semantic understanding, necessitates a comprehensive hardware/software co-design approach.

The remainder of this paper is organized as follows. Section II reviews related works in LLM optimization and real-time robotic systems, emphasizing techniques for efficient inference on embedded platforms. Section III presents our proposed hardware-software co-design methodology for implementing LLaMA 2 on an FPGA, detailing both the model compression pipeline and the accelerator architecture. Section IV introduces our experimental setup, reports quantitative results, and discusses performance trade-offs between embedded CPU, GPU, and FPGA deployments. Finally, Section V elaborates on the implications for HRI, provides insights into future directions in resource-constrained LLM-based robotics, concludes the paper, and outlines potential avenues for extending this work to larger-scale autonomous systems.

## II. Background & Literature Review

Numerous recent works illustrate the growing potential of LLMs in robotic applications. For instance, Wake et al. [19] demonstrated a chatbot system that integrates GPT-based speech generation with co-speech gestures, thereby enhancing user engagement in conversational settings. Similarly, Kannan et al. [20] proposed SMART-LLM (Smart Multi-Agent Robot Task Planning using LLMs), leveraging few-shot prompts to enable multi-robot task planning and successfully handling diverse task complexities in both simulation and real-world conditions. In the context of audio-guided navigation systems, Sun et al. [21] proposed TrustNavGPT, a method designed to model uncertainty in spoken instructions, thereby enhancing decision-making processes and improving resistance to adversarial commands.

Another line of research highlights the path-planning capabilities of LLMs. Latif [22] explored 3P-LLM, a GPT-3.5–turbo–based planner that outperforms classic algorithms in certain configurations thanks to its language-driven reasoning, despite continuing challenges in geometric accuracy. Meanwhile, Qi et al. [23] offered a comprehensive overview of integrating LLMs into robotic systems, outlining both task-oriented (e.g., multi-object rearrangement, navigation) and model-oriented applications, and proposing a robot intelligence framework that unifies natural language understanding with the Robot Operating System (ROS). Beyond navigation, Latif et al. [24] introduced PhysicsAssistant, a multimodal robot designed for K-12 science education, which combines GPT-3.5–turbo with computer vision to provide real-time support in laboratory investigations. Finally, Wang et al. [25] surveyed the application of multimodal LLMs in robotics, discussing how visual-language understanding can further broaden the scope of human-robot collaboration.

Taken together, these studies underscore how advanced LLMs can transform real-time guidance for the elderly, visually impaired, and broader assistive scenarios, yet they also highlight significant computational overhead that must be addressed for practical, on-device deployment.

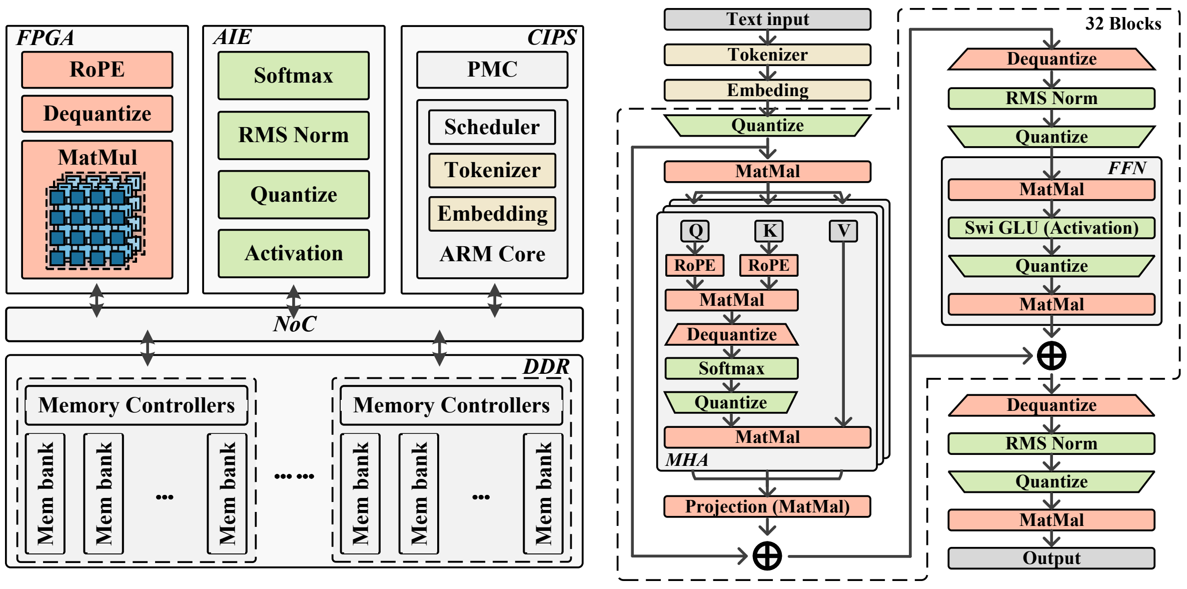

Fig. 2. Overview of the proposed FPGA-based system architecture.

In parallel, compute-efficient solutions for LLMs have gained attention, with approaches like Post-Training Quantization (PTQ), knowledge distillation, and specialized hardware acceleration. Notably, SpQR [26], GPTQ [27], and Additive Quantization [28] drive down memory footprints to 2–4 bits per parameter with minimal accuracy loss. Various teams push beyond GPUs to FPGAs for real-time, low-power inference. FlightLLM [29] and HLSTransform [30] map LLaMA 2-style architectures onto FPGAs for high energy efficiency, whereas LLaMAF [31] demonstrates pipeline-based quantized inference on an embedded FPGA. Hasan [32] further compares PTQ versus Quantization-Aware Training (QAT), emphasizing that balanced quantization can reduce the model size by over 60% while retaining acceptable performance. Table I highlights the overview of the compute-efficient solutions detailed previously. All inference speeds are reported in tokens per second (TPS). Where multiple model sizes or configurations are tested, and "N/A" indicates data not reported in the respective paper.

Despite these breakthroughs, robust co-design, which combines model compression, hardware-based parallelism, and real-time robotic control, remains underexplored. The remainder of this paper addresses this gap: We propose a hardware-software co-optimization framework for deploying LLMs on a quadruped robot platform, emphasizing resource reduction through FPGA acceleration and adaptive low-bit quantization while preserving advanced HRI functionalities.

## III. Methodology

To match the constraints of deploying LLMs on robots, we optimized the LLM algorithm with hardware and software co-design and deployed our design on a heterogeneous FPGA platform. The following subsections detail the techniques to enhance our FPGA implementation, ensuring it meets performance and energy efficiency requirements.

### A. LLM Optimization

In mobile robotics applications, more efficient algorithms lead to lower power consumption and extended operational time, particularly for computationally intensive models such as LLMs. For instance, the LLaMA 2 algorithm, which employs Group Query Attention (GQA), significantly reduces computational demands by operating with a smaller vocabulary, making it more suitable for deployment in robotic systems. The most computationally intensive component of LLaMA 2 is the transformer decoder, as shown in (1) and (2). GQA optimizes this by reducing the size of matrix weights $W_K$ and $W_V$, as well as the dimensions of key $\mathcal{K}$ and value $\mathcal{V}$ matrices, thereby lowering computational complexity in subsequent *Attention* calculations. Furthermore, we incorporate the *KV* cache to eliminate redundant computations of the $\mathcal{K}\mathcal{V}$ matrix, and further reduce algorithmic overhead using a sliding window mechanism since the control command context length of robots is fairly short. In our design, we constrain the sliding window size to further optimize computational efficiency in LLaMA 2.

$$Q = \text{RoPE}(XW_Q), \mathcal{K} = \text{RoPE}(XW_K), \mathcal{V} = XW_v \quad (1)$$

$$Attention(Q, \mathcal{K}, \mathcal{V}) = \text{SoftMax}(Q\mathcal{K}^T)\mathcal{V} \quad (2)$$

$$H = \text{Attention}_\text{i}(Q, \mathcal{K}, \mathcal{V}), \text{ where } i \in [1, h] \quad (3)$$

$$O = \text{MultiHead}(Q, \mathcal{K}, \mathcal{V}) = \text{Concat}(H_1, ..., H_h)W_P \quad (4)$$

### B. LLM Quantization

Quantization is a widely used technique for deploying LLMs, as it reduces computational complexity and lowers runtime memory consumption. However, global quantization can degrade model accuracy, particularly in low-bit settings. To mitigate the latter, we employ a group quantization approach, which balances computational efficiency and model accuracy.

As shown in (5) to (7), given a weight matrix $\mathbf{W}$, the group quantization method partitions $\mathbf{W}$ into $N$ smaller matrices $\{\mathbf{G_1}, \mathbf{G_2}, \ldots, \mathbf{G_N}\}$. A scaling factor $s_i$ is determined from the maximum absolute value $w_{\max}^{(i)}$ in each group $\mathbf{G_i}$, after which each weight in the group is quantized into an integer representation alongside a floating-point scaling factor:

$$\{\mathbf{G_1}, \mathbf{G_2}, \ldots, \mathbf{G_N}\} = \mathbf{W} \tag{5}$$

$$s_i = \frac{w_{\max}^{(i)}}{2^{(b-1)} - 1}, \text{ where } w_{\max}^{(i)} = \max(|\mathbf{G}_i|) \tag{6}$$

$$Q(\mathbf{G_i}) = \text{round}\left(\frac{\mathbf{G_i}}{s_i}\right) \tag{7}$$

$$\mathbf{W^q} = \{\mathbf{G_1^q}, \mathbf{G_2^q}, \ldots, \mathbf{G_N^q}\} \tag{8}$$

$$Size(\mathbf{W^q}) = \frac{b}{32} * Size(\mathbf{W}) + N * 4 \tag{9}$$

In matrix multiplications, the integer parts of the inputs are first multiplied to generate a double-bit-width integer, while the corresponding scaling factors are also multiplied to obtain the final output. The resulting double-bit-width integer is then restored to its original bit-width using an efficient hardware implementation trick.

This quantization scheme reduces quantization error to below 0.3%. However, certain LLM layers, such as normalization, softmax, and activation functions, are highly sensitive to weight precision, making even small errors unacceptable. To address this, our design employs a mixed-precision strategy, where group quantization is applied to matrix multiplications while floating-point operations are preserved for sensitive layers. This hybrid approach ensures both high model accuracy and reduced power consumption.

### C. Hardware Architecture

To further enhance computational efficiency, we design a heterogeneous computing platform for mixed-precision inference, as illustrated in Fig. 2. The platform distributes different computational tasks across specialized processing units: Rotary Position Embedding (RoPE), activation functions, and matrix multiplication modules are implemented on the FPGA, while Softmax, RMSNorm, and Quantization-Dequantization modules are handled by the Artificial Intelligence Engine (AIE). The Tokenizer and Embedding modules run on the CPU.

The matrix multiplication module operates on INT8 weights and inputs, utilizing corresponding group scaling factors. The results of multiplying two INT8 matrices are extended to INT16. To optimize precision and bandwidth usage, we discard the $x$ LSBs of the output while proportionally amplifying the scaling factor by $2^x$. This approach enables us to maintain perplexity comparable to 8-bit weights and 16-bit activations while operating entirely with 8-bit weights and activations, significantly reducing memory bandwidth consumption for intermediate results transferred between DDR memory and computation units distributed across the heterogeneous platform.

Although the quantized LLaMA 2 model significantly reduces memory requirements, the weight files remain too large for on-chip memory. Thus, all weights are stored in DDR memory and dynamically streamed into computational modules via a high-bandwidth Network-on-Chip (NoC) interface. Additionally, RoPE computations involve resource-intensive operations, such as floating-point power calculations for embedding values. To optimize FPGA resource usage, we precompute these values and access them via a LUT at runtime, improving performance while reducing computational overhead. For floating-point computations, this design accelerates RMSNorm and Softmax using a high-performance AIE matrix processor. Finally, the CPU orchestrates the execution of all modules, ensuring maximum computational efficiency across the heterogeneous architecture.

## IV. Experiments and Evaluation

### A. Evaluation Setup

**Models and Constraints:** We use LLaMA 2 as the primary test model, while our platform also supports other LLaMA variants such as TinyLLaMA and ShearedLLaMA. For quantization, we adopt INT8 grouped quantization for weights to enable comparisons with embedded CPU and GPU platforms. Additionally, we evaluate lower-bit quantization schemes down to INT4. In real-time robot interaction scenarios, input tokens primarily consist of state data, sensor readings, and task instructions, requiring shorter sequences but rapid response times. To optimize computational efficiency, we constrain the maximum input token length to 1024 and set the sliding window cache size to 50, reducing computational overhead while maintaining performance.

**CPU and GPU Baselines:** For CPU-based evaluations, we use an Intel® Xeon® Silver 4310 as a baseline, while for GPU-based evaluation, we deploy Nvidia® Jetson AGX Orin™ (64 GB). Detailed hardware configurations are provided in Table II. We analyze CPU performance using the C++ implementation of LLaMA 2 and monitor power consumption via the `powerstat` tool. For the Jetson AGX Orin™ platform, we deploy LLaMA 2 using `text-generation-webui`, an official tool provided by Nvidia®, and measure power consumption using the Jetson™ Power GUI tool.

**FPGA Implementation:** Our FPGA-based implementation is deployed on the AMD® Versal™ VCK190 evaluation platform, with an overview shown in Fig. 2. Since the FPGA logic performs INT8 arithmetic, while the AIE executes floating-point arithmetic, we integrate quantization and dequantization modules to facilitate data transfer between these components. The AIE consists of 2D arrays of AI Engine tiles and DSPs, interconnected via a high-bandwidth NoC, and operates at a clock frequency exceeding 1 GHz. To ensure a fair comparison across platforms, we constrain the FPGA logic and AIE modules to a uniform clock frequency of 200 MHz.

### B. Performance Evaluation and Analysis

We evaluate the impact of quantization algorithms and hardware platforms using LLaMA 2 as a benchmark. As shown in Fig. 3, we assess the quantization results for INT4 and INT8 bit-widths. Since only the matrix multiplication weights are quantized to maximize model accuracy, the overall weight size

TABLE II

Comparison of LLaMA 2-7B implementation across different embedded platforms

| Platform | Memory (Bandwidth) | Frequency | Computing Units | Latency | Throughput | Power | Efficiency |
|---|---|---|---|---|---|---|---|
| Xeon Silver 4310 | 377GB DDR4 (25.6GB/s) | 2100 MHz | 12 CPU cores | 1.1 s | 5.16 TPS | 120 W | 0.04 token/J |
| Jetson AGX Orin | 64GB LPDDR5 (204.8 GB/s) | 1400 MHz | 2048 CUDA cores | 1.51 s | 16.07 TPS | 35 W | 0.9 token/J |
| U280 [29] | 8GB HBM + 44MB on-chip memory | 225 MHz | 3840 DSPs | - | 55 TPS | 45 W | 1.22 token/J |
| Ours | 8GB DDR4 + 70MB on-chip memory | 200 MHz | 1968 DSPs | 0.7 s | 31.29 TPS | 18 W | 1.73 token/J |

for INT8 quantization is not exactly one-fourth of the original. However, INT4 quantization achieves a 47% reduction in model size compared to INT8.

Additionally, we evaluate the effect of quantization group size on model compression. For the same precision level, reducing the group size from 128 to 64 results in a 6% increase in weight size. We further analyze the trade-off between quantization parameters and model perplexity on the Wikitext dataset. Our results indicate that INT8 group quantization increases model complexity by 5% compared to the original model. Furthermore, perplexity continues to degrade as the quantization bit-width is reduced.

Despite the small error introduced by group quantization (0.4% at most), increasing the group size has minimal impact on model complexity while significantly reducing parameter count, thereby improving computational performance.

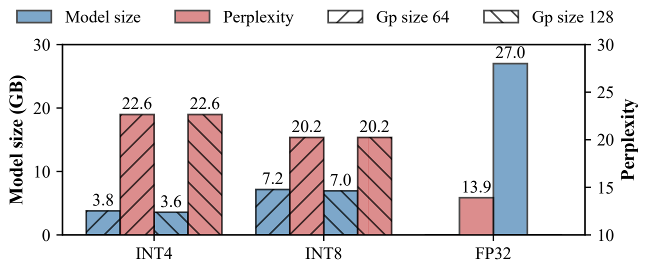

Fig. 3. Comparison of model size and perplexity across different quantization precisions.

Looking into the yielded performance, and as observed in Table II, we maintain the same precision level across CPU, GPU, and FPGA comparisons. For INT8 quantization, the CPU inference speed is 5.17 TPS, while the Jetson AGX Orin achieves 16.07 TPS. In comparison, our FPGA implementation achieves 6.06× and 1.95× higher throughput, respectively.

In robotic applications, particularly in HRI, the LLM throughput requirement is mainly dictated by human reading and speaking speeds. As long as the LLM throughput exceeds the speaking rate, it meets real-time interaction requirements. However, beyond throughput, HRI performance is more sensitive to response latency, which refers to the time delay between input reception and the start of model output. Our design, leveraging a streaming compute engine and a high-performance on-chip network, achieves an output latency of less than 400 ms.

Additionally, power consumption plays a critical role in the runtime of robotic systems. Our FPGA implementation achieves 1.92× more inference calls compared to Jetson AGX Orin for the same power consumption, demonstrating superior efficiency in robotic control. Even when compared with the state-of-the-art FlightLLM [29] implementation, which utilizes cloud-based FPGA hardware (U280), our design achieves a 30% improvement in computational efficiency.

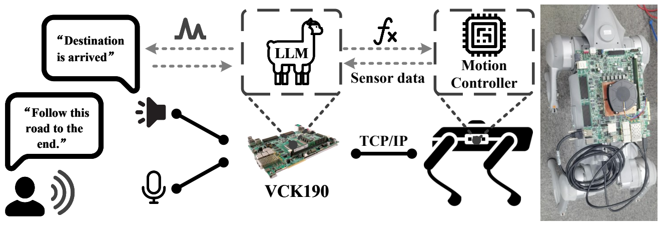

Fig. 4. A computationally efficient robotic system for real-time assistive applications.

## C. Real-World Deployment and Evaluation

To assess the performance of our locally accelerated LLM solution in real-world robotic applications, we deployed our design on a Unitree Go2 integrated with an AMD Versal VCK190 platform. As illustrated in Fig. 4, voice commands are transcribed into text using an intelligent microphone and combined with real-time sensor data from Go2 (e.g., LiDAR) to form the LLM input.

The processed LLM output consists of two key components:

1) HRI feedback; providing real-time responses to the user.
2) Motion control commands for Go2, which are finally sent to the motion controller, ensuring adherence to safety constraints.

For comparison, we conducted the same experiment using Jetson AGX Orin and also the remote server, where the outputs were split and transmitted to their respective targets. Our results indicate that the server-based LLM exhibits similar inference latency (1.3 s) to CPU baselines, but introduces an additional 100ms of network transmission delay. The LLM inference latency remains the dominant factor in the overall system response time.

Ultimately, our FPGA-based implementation achieves significant latency improvement over the server-based solution the Jetson deployment, demonstrating superior efficiency for real-time robotic applications.

## V. Conclusion

This paper presents an FPGA-based optimization technique for deploying LLMs on resource-constrained robotic systems, achieving up to a 77% reduction in computational latency while preserving real-time interaction. Deploying a LLaMA 2-7B model on a Unitree Go2 robotic dog demonstrates significantly higher throughput than CPU and GPU baselines, with a 30% improvement over state-of-the-art FPGA implementations. By combining GQA, group quantization, and FPGA acceleration, we minimize LLaMA 2-7B's inference cost while maintaining dialogue fluency and decision-making for assistive robotics. These results highlight the potential of FPGA-accelerated LLMs for efficient, scalable human-robot interaction in dynamic environments. Future work will explore the proposed techniques as an all-in-one system to further enhance efficiency and generalizability across diverse robotic platforms.

## References

[1] OpenAI, *et al.*, "GPT-4 technical report," 2024. [Online]. Available: https://arxiv.org/abs/2303.08774

[2] J. Devlin, *et al.*, "BERT: Pre-training of deep bidirectional transformers for language understanding," 2019. [Online]. Available: https://arxiv.org/abs/1810.04805

[3] H. Touvron, *et al.*, "LLaMA: Open and efficient foundation language models," 2023. [Online]. Available: https://arxiv.org/abs/2302.13971

[4] Z. Wang, *et al.*, "FederatedScope-GNN: Towards a unified, comprehensive and efficient package for federated graph learning," in *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, ser. KDD '22. ACM, Aug. 2022, p. 4110–4120. [Online]. Available: http://dx.doi.org/10.1145/3534678.3539112

[5] T. M. Al-Hasan, *et al.*, "From traditional recommender systems to GPT-based chatbots: A survey of recent developments and future directions," *Big Data and Cognitive Computing*, vol. 8, no. 4, p. 36, Mar. 2024. [Online]. Available: http://dx.doi.org/10.3390/bdcc8040036

[6] J. Liu, *et al.*, "Retrieval-based knowledge transfer: An effective approach for extreme large language model compression," in *Findings of the Association for Computational Linguistics: EMNLP 2023*, H. Bouamor, *et al.*, Eds. Singapore: Association for Computational Linguistics, Dec. 2023, pp. 8643–8657. [Online]. Available: https://aclanthology.org/2023.findings-emnlp.578/

[7] M. Z. Karimi, *et al.*, "Enhanced accessibility for visually impaired & blind artists (eaviba) using machine learning," *Int. J. Sci. Res. Eng. Manag.*, vol. 08, no. 05, p. 1–5, May 2024. [Online]. Available: http://dx.doi.org/10.55041/ijsrem32878

[8] O. Ahia, *et al.*, "The low-resource double bind: An empirical study of pruning for low-resource machine translation," in *Findings of the Association for Computational Linguistics: EMNLP 2021*, M.-F. Moens, *et al.*, Eds. Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 3316–3333. [Online]. Available: https://aclanthology.org/2021.findings-emnlp.282/

[9] G. Yang, *et al.*, "Dynamic stashing quantization for efficient transformer training," 2023. [Online]. Available: https://arxiv.org/abs/2303.05295

[10] H. Liu, *et al.*, "Enhancing the LLM-based robot manipulation through human-robot collaboration," *IEEE Robot. Autom. Lett.*, vol. 9, no. 8, p. 6904–6911, Aug. 2024. [Online]. Available: http://dx.doi.org/10.1109/LRA.2024.3415931

[11] J. K. DeHart, "Leveraging large language models for direct interaction with SysML v2," *INCOSE International Symposium*, vol. 34, no. 1, p. 2168–2185, July 2024. [Online]. Available: http://dx.doi.org/10.1002/iis2.13262

[12] H. Zhou, *et al.*, "LLM-BT: performing robotic adaptive tasks based on large language models and behavior trees," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, May 2024, p. 16655–16661. [Online]. Available: http://dx.doi.org/10.1109/ICRA57147.2024.10610183

[13] T. J. Ham, *et al.*, "A^3: Accelerating attention mechanisms in neural networks with approximation," in *2020 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 2020, pp. 328–341.

[14] E. Yvinec, *et al.*, "SPIQ: Data-free per-channel static input quantization," in *2023 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. IEEE, Jan. 2023, p. 3858–3867. [Online]. Available: http://dx.doi.org/10.1109/wacv56688.2023.00386

[15] A. Vishnu Kadam, "Text-to-speech in voice assistants: Challenges and mitigation strategies," *J. Eng. Appl. Sci. Technol.*, p. 1–4, Mar. 2023. [Online]. Available: http://dx.doi.org/10.47363/jeast/2023(5)183

[16] J. Kwon, *et al.*, "Sparse convolutional neural network acceleration with lossless input feature map compression for resource-constrained systems," *IET Comput. Digit. Tech.*, vol. 16, no. 1, p. 29–43, Nov. 2021. [Online]. Available: http://dx.doi.org/10.1049/cdt2.12038

[17] S. Dave, *et al.*, "Hardware acceleration of sparse and irregular tensor computations of ML models: A survey and insights," *Proceedings of the IEEE*, vol. 109, no. 10, pp. 1706–1752, 2021.

[18] S. Lin, *et al.*, "Efficient micro-structured weight unification and pruning for neural network compression," 2021. [Online]. Available: https://arxiv.org/abs/2106.08301

[19] N. Wake, *et al.*, "GPT models meet robotic applications: Co-speech gesturing chat system," 2023. [Online]. Available: https://arxiv.org/abs/2306.01741

[20] S. S. Kannan, *et al.*, "Smart-LLM: Smart multi-agent robot task planning using large language models," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2024, pp. 12 140–12 147.

[21] X. Sun, *et al.*, "TrustNavGPT: Modeling uncertainty to improve trustworthiness of audio-guided LLM-based robot navigation," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2024, pp. 8794–8801.

[22] E. Latif, "3P-LLM: Probabilistic path planning using large language model for autonomous robot navigation," 2024. [Online]. Available: https://arxiv.org/abs/2403.18778

[23] Z. Qi and X. Jing, "Advances in large language models for robotics," in *2024 7th International Conference on Mechatronics, Robotics and Automation (ICMRA)*. IEEE, Sept. 2024, p. 72–76. [Online]. Available: http://dx.doi.org/10.1109/ICMRA62519.2024.10809099

[24] E. Latif, *et al.*, "PhysicsAssistant: An LLM-powered interactive learning robot for physics lab investigations," in *2024 33rd IEEE International Conference on Robot and Human Interactive Communication (ROMAN)*. IEEE, Aug. 2024, p. 864–871. [Online]. Available: http://dx.doi.org/10.1109/RO-MAN60168.2024.10731312

[25] J. Wang, *et al.*, "Large language models for robotics: Opportunities, challenges, and perspectives," *J. Autom. Intell.*, Dec. 2024. [Online]. Available: http://dx.doi.org/10.1016/j.jai.2024.12.003

[26] T. Dettmers, *et al.*, "SpQR: A sparse-quantized representation for near-lossless LLM weight compression," 2023. [Online]. Available: https://arxiv.org/abs/2306.03078

[27] E. Frantar, *et al.*, "GPTQ: Accurate post-training quantization for generative pre-trained transformers," *arXiv preprint arXiv:2210.17323*, 2022.

[28] V. Egiazarian, *et al.*, "Extreme compression of large language models via additive quantization," in *International Conference on Machine Learning*. PMLR, 2024, pp. 12 284–12 303.

[29] S. Zeng, *et al.*, "FlightLLM: Efficient large language model inference with a complete mapping flow on FPGAs," in *Proceedings of the 2024 ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, 2024, pp. 223–234.

[30] D. Y. Key, *et al.*, "HLSTransform: Energy-efficient LLaMA 2 inference on FPGAs via high level synthesis," in *Workshop on Efficient Systems for Foundation Models II @ ICML2024*, 2024. [Online]. Available: https://openreview.net/forum?id=tvmCsGnyLq

[31] H. Xu, *et al.*, "LLaMaf: An efficient LLaMA2 architecture accelerator on embedded FPGAs," in *2024 IEEE 10th World Forum on Internet of Things (WF-IoT)*. IEEE, 2024, pp. 1–7.

[32] J. Hasan, "Optimizing large language models through quantization: A comparative analysis of PTQ and QAT techniques," 2024. [Online]. Available: https://arxiv.org/abs/2411.06084