VASAVI COLLEGE OF ENGINEERING
IBRAHIMBAGH, HYDERABAD

IMPLEMENTATION OF LARGE LANGUAGE MODEL ON FPGA

A PROJECT REPORT

Submitted By:

1602-22-735-099 Yennam Sai Tharun Reddy

1602-22-735-118 P.Supriya Sree

1602-22-735-103 Uduthanaboina Sathwik

Submitted To:

Department of Electronics & Communication Engineering

Vasavi College of Engineering

Imbrahimbagh, Hyderabad

500031

December, 2024

DECLARATION

We, the undersigned, declare that the project report **Implementation of Large Language Model on FPGA** submitted for partial fulfillment of the requirements for the award of the degree of Bachelor of Engineering of Vasavi College of Engineering(A), Telangana, is a work done by me under the supervision of Dr. Vibha Kulakarni, Mr. V. Krishna Mohan. I hereby affirm that this submission is a true representation of my own ideas and words. In instances where I have incorporated the ideas or words of others, I have diligently and accurately cited and referenced the sources. Furthermore, I confirm my adherence to the principles of academic honesty and integrity, ensuring that no data, idea, fact, or source has been misrepresented or fabricated in my submission. I fully comprehend that any deviation from the aforementioned standards may result in disciplinary action by the institute and/or the University, and may also prompt legal action from individuals whose work has not been appropriately cited or for which proper permission has not been obtained. Additionally, I declare that this report has not been previously used as the basis for the award of any degree, diploma, or similar title from another University.

Name : Y.Sai Tharun Reddy
Signature:.....
Name : P.Supriya Sree
Signature:.....
Name : U.Sathwik
Signature:.....

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

Vasavi College of Engineering

Ibrahimbagh,Hyderabad

500031

CERTIFICATE

This is to certify that the report entitled **IMPLEMENTATION OF LARGE LANGUAGE MODEL ON FPGA** Submitted by **Y.Sai Tharun Reddy,P.Supriya Sree,U.Sathwik** to the Vasavi College of Engineering in partial fulfillment of the requirements
for the award of the Degree of Bachelor of Technology in Electronics & Communication Engi- neering is a bonafide record of the project work carried out by him/her under my/our guidance and supervision. This report in any form has not been submitted to any other University or In- stitute for any purpose.

Project Coordinator:
Name : Dr. Vibha.K
Signature:.....

Project Coordinator:
Name : Mr.V.Krishna Mohan
Signature:.....

Head of Department
Name : Dr.E Sreenivasa Rao
Signature:.....

ACKNOWLEDGEMENT

I wish to record my indebtedness and thankfulness to all who helped me to prepare this Project titled **LARGE LANGUAGE MODEL ON FPGA** and present it satisfactorily

I am especially thankful for my guides Dr .Vibha Kulakarni and Mr. V.Krishna Mohan in the Department of Electronics & Communication of Engineering for giving me valuable suggestions and critical inputs in the preperation of this report. I am also thankful to Dr.E Sreenivasa Rao, Head of Department of Electronics & Communication of Engineering for encouragement

Thank you all for your assistance and support.

Y. sai Tharun Reddy
P. Supriya Sree
U. Sathwik
B.E(Elec
tronics & Communication of Engineering)
Department of Electronics & Communication of Engineering
Vasavi College of Engineering

Abstract

Large language model (LLM) deployment across heterogeneous hardware plat- forms is a major obstacle to modern artificial intelligence (AI) research and practice. Even while LLMs have proven to be unmatched in their ability to comprehend and generate natural language, there is still significant concern regarding their effective deployment and use on a variety of hardware architectures, such as CPUs, GPUs, and FPGAs.

It is imperative to optimize LLMs for various hardware configurations in order to guarantee scalability, performance, affordability, and energy efficiency. In addi- tion, it is essential to tackle the challenges associated with implementing LLMs on heterogeneous hardware platforms in order to facilitate edge computing, real-time applications, and universal access to cutting-edge AI technologies in various settings and user bases.

Thus, the core of our research problem is to comprehend the nuances of LLM deployment on various hardware architectures, with the goal of realizing the full po- tential of these transformational models while tackling the pragmatic issues related to their use and implementation.

TABLE OF CONTENTS

1. INTRODUCTION

According to this paper, large language models (LLMs), including the well-known Transformer-based models, have revolutionized a number of fields by demonstrating exceptional performance on tasks involving the generation and processing of natural language. However, because these models have large compute and memory overheads,efficiently deploying them is difficult.

1.1. Efficiency Enhancement:

- Large computational resources are needed for inference with LLMs. FPGAs have the potential to speed up LLM inference by taking advantage of their parallelism and scalability.
- Using FPGA-specific resources like DSP48 blocks and heterogeneous memory, FlightLLM, for example, offers effective LLM inference on FPGAs.
- The objective is to outperform commercial GPUs in terms of cost and energy efficiency.

1.2. Motivation

Our motivation to work on this Testing large language models (LLMs) on various hardware platforms such as CPUs, GPUs, and FPGAs since we need to maximize energy efficiency, performance, and cost. Through the assessment of these heterogeneous platforms, we are able to determine the optimal configurations for different applications, guaranteeing that LLMs function at maximum efficiency. By using energy-efficient solutions, this strategy not only encourages sustainable AI practices but also increases accessibility and inclusivity for cutting-edge AI technologies. Furthermore, our research encourages innovation in software and hardware, setting us up for future developments and guaranteeing the continued stability, scalability, and versatility of our AI deployments.

1.3. Objectives

The goal of this research is to maximize the use of large language models (LLMs) across a range of hardware platforms, including as FPGAs, GPUs, and CPUs. Among our goals are

- **Performance Optimization**: Assess LLM's performance on various hardware platforms to find the best setups for applications like text generation and classification.
- **Cost-Efficiency**: Examine cost-effectiveness by taking maintenance costs, energy usage, and hardware procurement into account.

- **Energy E*ffi*ciency**: Look into ways to minimize the impact on the environment and operating expenses by consuming less energy throughout LLM deployment.
- **Real-Time Applications**: Consider putting LLMs on edge devices for chatbots and voice assistants, among other real-time applications.
- **Inclusivity**: Make sure that advanced AI technologies are accessible to users with a range of abilities by ensuring accessibility across various hardware.

2. LITERATURE SURVEY

This section of our project report presents the research done for the project.

There has been a massive adoption of Large Language Models (LLMs) toward advanced computational tasks since they are capable of processing and generating human-like text. However, the strict need for deploying these models and enhancing their overall efficiency requires advanced hardware performance and good software compatibility. This is a continuation of work done in the previous semester, whereby performance analysis of LLMs on various types of hardware platforms was conducted on CPUs and GPUs. Within this, the computational requirements of LLMs and scalability were understood.

One of the main aspects of this semester's work included building a customized OS using PetaLinux, a versatile framework for building Linux distributions on embedded platforms. This framework reduces complexity in system configuration and kernel customization, thus producing boot images specifically made for the AMD VCK5000 Versal AI Core board. These boot images ensure that the FPGA is initialized and allows for the LLM inference pipeline to run without wasting any hardware resources. The customized OS not only streamlines the deployment process but also helps at the software-software integration level, which is considered to be the basis of an efficient LLM inference system.

Having reviewed a number of FPGA platforms, the best hardware to be used for our application was the VCK5000 board. The architecture of this board combines the AI engines, the programmable logic, and the high-speed interfaces to give it an edge over many other boards in computationally intensive AI tasks. This board supports tools like Vitis AI and PetaLinux, which make the compilation and optimization of AI models in streaming environments a lot easier to deploy. The VCK5000 had a good advantage of handling tasks with large throughput at the cost of minimizing power consumption in the selection criteria, matching the performance and efficiency goals for the project. It was through cloud and edge computing involvement that LLM deployment was enabled. Although cloud computing offers scalability to resources required for training and large-scale inference, edge computing enables real-time processing with minimal latency and bandwidth usage.

Deploying LLMs on FPGA-based edge devices like the VCK5000 accomplishes low-latency performance with reduced dependency on centralized cloud infrastructure. This hybrid approach reflects industry trends that balance the need for computational efficiency and responsiveness in distributed AI systems. This demonstrates an integrative software and hardware system deployment for LLMs, leveraging OS customization using PetaLinux, hardware acceleration on VCK5000, and insightful ideas borrowed from cloud and edge computing paradigms. Therefore, the system is primed to deliver high-performance inference for real-time AI applications. This work shows the potential application of FPGA-based solutions in improving efficiency and scalability in AI systems.

The literature also emphasizes how pivotal performance measures for assessing hardware platforms for LLMs are, including throughput, latency, energy efficiency, and cost-effectiveness. The most suitable hardware is chosen based on these characteristics for particular LLM operations.

To sum up, the field of implementing LLMs on various hardware platforms is dynamic and demands a balance between cost, energy usage, and performance. The creation of specialized hardware accelerators will be essential to facilitating LLMs' wider acceptance and incorporation into practical applications as they continue to expand in size and complexity. This survey sheds insight on the trade-offs and factors to consider while deploying these models on various platforms by offering a succinct summary of the state of hardware accelerators for LLMs.

3. DESIGN AND METHODOLOGY

- **XSA File Generation(Vivado)**
- **petalinux(OS)/cloud server**
- **Integration of ONNX Model**
- **IMPLEMENTATION ON FPGA**

4. Hardware-Software co-design

Hardware-software co-design represents a contemporary engineering strategy that merges the high-speed processing abilities of hardware with versatility and adaptability of software. This approach enhances computational efficiency, power conservation, and scalability, making it particularly suitable for demanding applications such as Large Language Models (LLMs) and gunshot detection systems. By intelligently distributing workloads between hardware and software, co-design maximizes resource efficiency and accommodates real-time processing needs.

Key Features of Hardware-Software Co-design:

System Partitioning:

The design process starts by dividing tasks between hardware and software. Tasks that are compute-heavy, repetitive, and suitable for parallel execution—such as matrix multiplications, convolutions, or are assigned to hardware. Meanwhile, tasks that require adaptability and control, including orchestration, model updates, or non-essential signal processing, are retained in the software realm.

Custom Hardware Accelerators:

Custom hardware accelerators are created for specific tasks like matrix multiplications for LLMs, activation functions like GELU or ReLU, and cross-correlation for signal processing in gunshot detection applications. Tools like Vitis HLS allow for the high-level synthesis of these accelerators, making the development process more straightforward.

Efficient Memory Management

FPGA designs are optimized for memory utilization by employing on-chip RAM alongside external DDR memory to store weights, activations, and intermediate data. Double buffering techniques facilitate continuous data transfer without interrupting computation.

Quantization and Precision Optimization

Low-precision computations (for instance, using INT8 for LLMs) are adopted to minimize memory usage and enhance processing speeds while maintaining acceptable levels of accuracy.

Parallel Processing and Pipelining

FPGAs are adept at processing multiple tasks in parallel, allowing various data streams to be handled at the same time. Pipelining also speeds up computation by overlapping different stages of processing.

Interface Management

AXI interfaces are used to manage communication between hardware and software components effectively. This allows for quick data transfers, resulting in low latency and high throughput.

Software Framework Integration

The co-design process seamlessly incorporates hardware accelerators into software frameworks like TensorFlow or PyTorch through Vitis AI-provided APIs. This integration facilitates the use of pre-trained models and speeds up deployment.

Scalability and Reconfigurability

The co-design methodology supports a modular structure, enabling the system to grow according to application needs. Furthermore, hardware reconfiguration allows for updates to the accelerators without necessitating a full redesign of the system.

Applications in LLMs

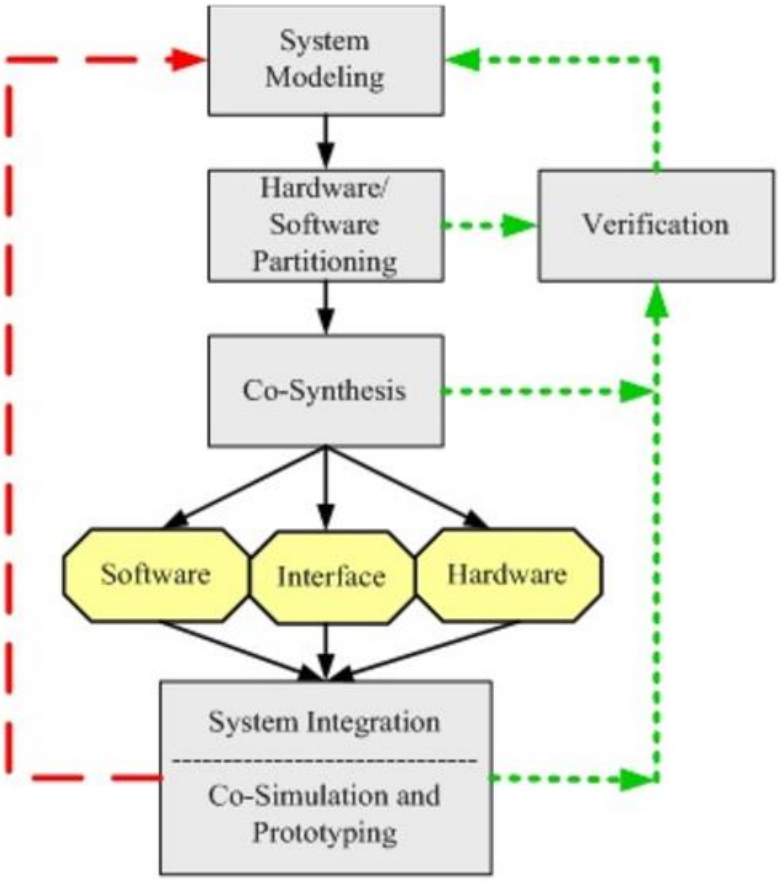
For Large Language Models, co-design takes advantage of FPGA accelerators to efficiently manage dense matrix multiplications and activation functions within transformer layers. Hardware handles attention mechanisms, including softmax and dot-product computations, for enhanced speed and parallelism. Meanwhile, the software oversees these operations, manages data flow, and ensures integration with frameworks like TensorFlow. Quantization techniques employed in hardware dramatically cut down on memory requirements, making LLM inference processes more efficient.

Development Workflow

- **System Design:** Identify which tasks are suitable for hardware acceleration.
- **Hardware Implementation:** Create accelerators using Vitis HLS with a focus on modular design and performance.
- **Software Integration:** Develop control logic and drivers to connect with hardware components.
- **Co-Verification:** Conduct hardware-in-the-loop (HIL) and software-in-the-loop (SIL) testing to ensure alignment.
- **Deployment:** Prepare FPGA bitstreams and software binaries for final integration.

Advantages of Hardware-Software Co-design

- **High Performance:** Offloading essential tasks to hardware accelerators decreases latency and boosts throughput.
- **Energy Efficiency:** FPGA logic consumes less power compared to CPUs and GPUs when performing parallel tasks.
- **Flexibility:** Software allows for easy updates, management, and control operations.
- **Real-Time Processing:** The synergy of FPGA's parallel capabilities and optimized memory access ensures efficient timing during operations.



5. VIVADO

Vivado is from Xilinx (just recently acquired by AMD) and is specifically used for developing, simulating, and programming digital systems onto FPGA devices. Being a successor to Xilinx ISE Design Suite, Vivado offers users a more modern, integrated development environment with enhanced capabilities

5.1. Key Features of Vivado:

1. Unified Design Environment: Integrate various aspects of FPGA design into one platform, as is true for the entry of a design, simulation, synthesis, implementation, and debugging. 2. High-Level Design Support: Enables designs that can be made using HLS tools to convert C, C++, and SystemC into HDL. Integrate with MATLAB and Simulink for system modelling. 3. Optimized for Large Designs Designed to support large, complex FPGA devices such as Xilinx UltraScale, Zynq, and Versal families. 4. Advanced Timing Analysis: Provides a powerful STA tool to check designs against timing requirements 5. Vivado Simulator: Has a simulation engine for functional and timing simulation for most applications which replaces the external simulators.

5.2. Vivado Editions:

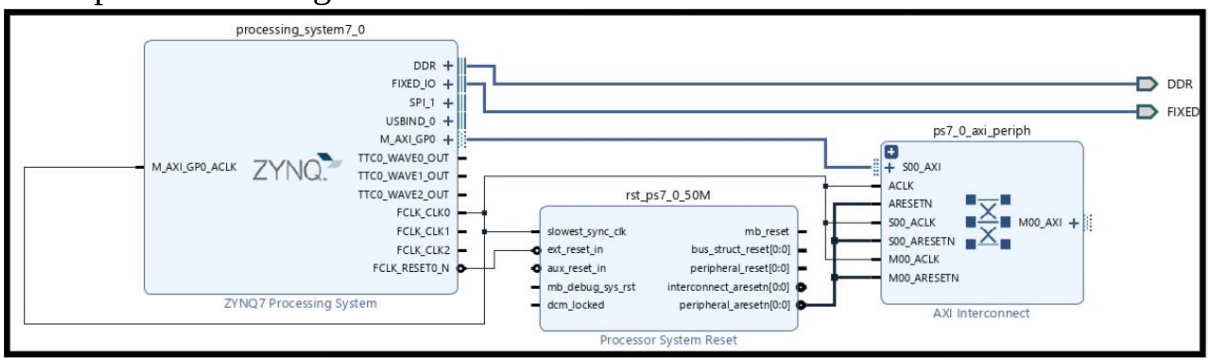
1. Vivado WebPACK (Free): Provides a light feature subset as the support to devices like Artix-7 and Spartan family. Ideal for newcomers, hobby enthusiasts, and small applications. 2. Vivado Design Edition (Licensed): Offers enhanced support for all Xilinx devices, high speed design along with advanced analysis tools. 3. Vivado System Edition (Licensed): All features from the Design Edition, in addition to some extra tools for system-level design, like High-Level Synthesis and DSP design tools.

5.3. Vivado's Major Components:

1. Project Manager: Manages project files, constraints, and IPs. 2. Schematic Editor: Tends to support visualization and editing of designs at the schematic level. 3. IP Catalog: A library of pre-designed and tested IP cores-a place to find communication protocols, as well as DSP blocks. 4. Constraint Manager: Deals with design constraints, including pin assignments, timing constraints, and power requirements. 5. Implementation Tools: Includes synthesis, placement, routing, and bitstream generation

5.4. XSA:

XSA refers to Xilinx Shell Archive, which is a file format used in Xilinx Vivado in order to encapsulate both the hardware designs and configuration details of an FPGA or SoC project. It acts as one of the key inputs in the generation of embedded software or integrating hardware designs with the tools for software like Xilinx Vitis. FPGA hardware design (bitstream or partial bitstream). It acts as a bridge between the hardware design process in Vivado and the software development process in Vitis or Peta Linux. Facilitates the transition from hardware design to software development. In Vivado during the export process of the hardware the synthesized design is packaged with pertinent configuration files.



6. PetaLinux

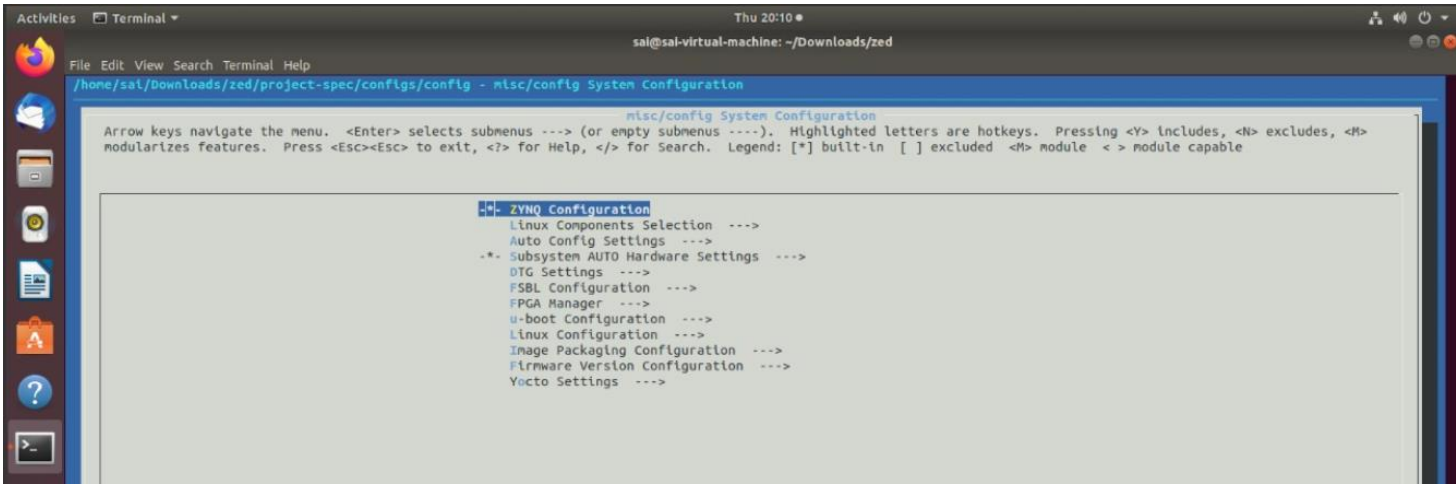
PetaLinux is a collection of development tools created to simplify the creation, modification, and deployment of embedded Linux systems on Xilinx hardware platforms, including FPGAs and SoCs. It offers a comprehensive environment for configuring the Linux kernel, developing boot loaders, device drivers, and file systems specifically designed for embedded devices. Yocto-based workflows are supported by PetaLinux, allowing users the flexibility to customize software while integrating various open-source libraries and tools. Support for hardware-accelerated processing, pre-configured templates for frequently used applications, powerful debugging tools, and seamless integration with Xilinx's Vivado Design Suite are some of its salient features.

PetaLinux is an important tool for implementing Large Language Models (LLMs) on FPGAs. FPGAs are increasingly being used to accelerate machine learning workloads because of their ability to execute parallel computations efficiently while remaining energy efficient. PetaLinux enables the implementation of customized embedded Linux systems that control FPGA hardware resources and facilitate the integration of LLM frameworks. It allows the use of hardware accelerators such as AI Engines or custom circuitry to optimize important operations in LLMs such as matrix multiplications, attention mechanisms, and transformer-based calculations. Using PetaLinux, developers can fine-tune Linux kernels to achieve optimal hardware-software co-design, reducing latency and increasing throughput for real-time LLM applications.

Key Features of PetaLinux

- **Kernel and Device Tree Customization**: Tailors the Linux kernel and device tree to specific hardware configurations.
- **Hardware Acceleration Support**: Works with the FPGA fabric to utilize hardware accelerators for computationally intensive operations.
- **Wide Peripheral Support**: Includes drivers and libraries for a variety of interfaces such as UART, SPI, I2C, Ethernet, and others.
- **Yocto Integration**: Allows for flexible package management and customization using Yocto recipes.
- **Debugging and Monitoring Tools**: Includes runtime debugging tools such as GDB, kernel log analyzers, and profiling tools.
- **SDK Generation**: Offers cross-compilation toolchains for application development.

PetaLinux additionally features peripheral interfaces that enable seamless data flow between the FPGA and external systems. This is extremely important in edge computing applications, where FPGA-based LLMs communicate with sensors, network devices, or cloud-hosted systems. PetaLinux's complete toolchain simplifies the management of complex hardware configurations, making it an effective enabler for implementing LLMs on FPGAs in applications that require low-power, high-performance, and versatile solutions.



7. vitis AI

AMD Vitis AI is a cutting-edge AI development platform designed to accelerate deep learning inference on hardware such as FPGAs (Field-Programmable Gate Arrays) and ACAPs (Adaptive Compute Acceleration Platforms). It offers unparalleled flexibility, decreased latency, enhanced performance, and efficiency in creating and deploying AI applications on edge and data center platforms.

Exclusive Features of Vitis AI:

1. Pre-trained models and optimization of models:

It provides access to libraries of pre-trained models that are highly optimized for AMD's hardware and can be used for latest technologies like natural language processing and computer vision.It includes techniques such as quantization,pruning and compression of hte AI models to ensure that there's an optimized performance and efficiency.

2. Hardware Acceleration:

The FPGAs and deep learning processor units developed by AMD are utilized to deliver high-throughput AI inference with low latency.

3.Std Framework Compliance:

The platform is compatible with frameworks including TensorFlow, PyTorch, and Caffe that enables the integrations of these pre-existing AI models.

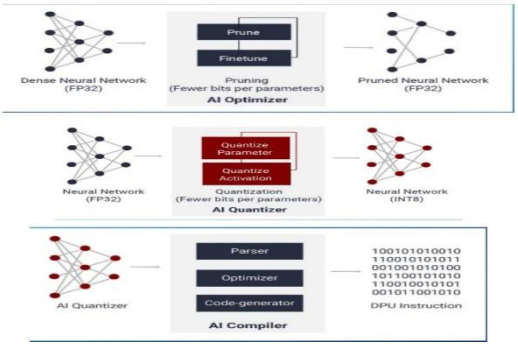
Benefits

Cost and Energy Efficiency: In comparison with conventional GPUs, FPGA solutions consume less energy and also cost less money.

Customizability: Total control over hardware and software upgrades.

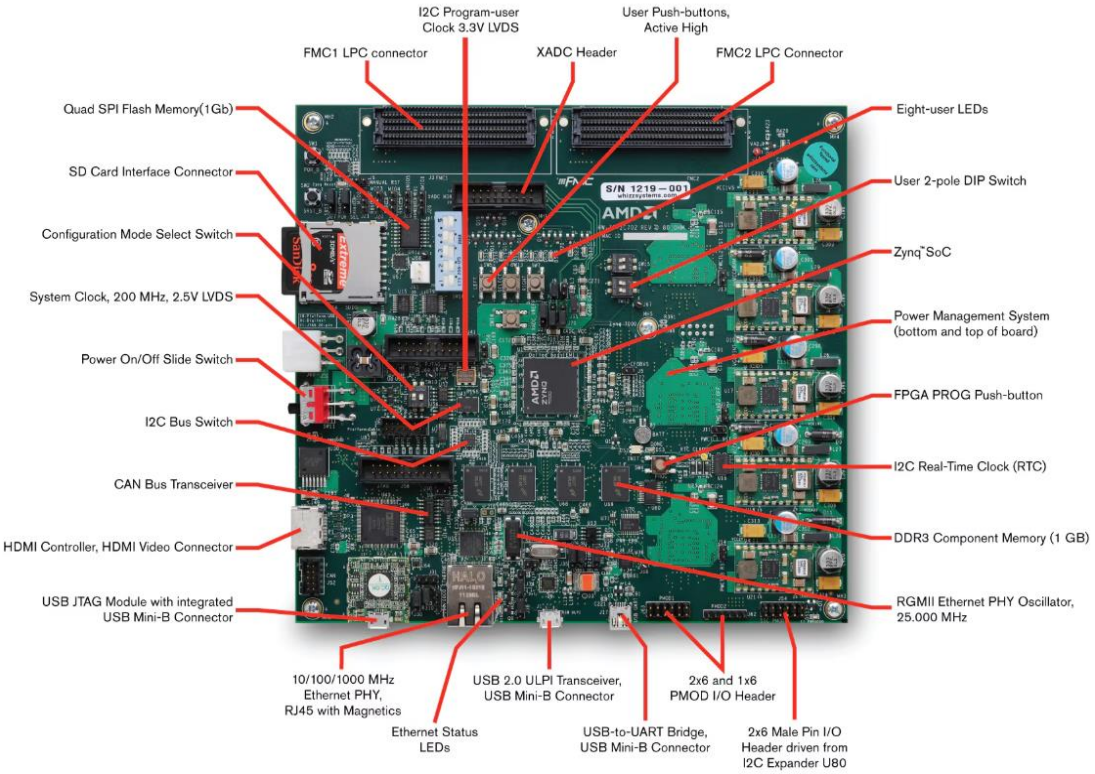
Real-Time Processing: Minimum latency is required for such mission-critical applications as Internet of Things and driverless cars.

Scalability: Well-fitted for cloud and edge settings.



8. HARDWARE

Zedboard: The ZedBoard (Zynq Evaluation and Development Board) is a highly versatile development platform designed around the Xilinx Zynq-7000 All Programmable SoC. It integrates a dual-core ARM Cortex-A9 processor with FPGA programmable logic, making it ideal for embedded system design, hardware-software codesign, and rapid prototyping. Zynq-7000 SoC Processing System (PS): Includes a dual-core ARM Cortex-A9 processor (running up to 1 GHz), providing a software development platform. Programmable Logic (PL): FPGA fabric enables custom hardware accelerators, signal processing, or any application-specific hardware design. Development Workflow Design the Hardware: Use Vivado to create a hardware design in the FPGA fabric. Envelop the Software: Use Vitis or an SDK to program the ARM processor for application logic. Integrate Hardware and Software: Create a seamless interface between the ARM processor and custom hardware logic in the FPGA. Deploy and Debug: Use tools like UART and onboard LEDs for debugging and testing.



VCK 5000: The VCK5000 Versal AI Core Board is an Adaptive Compute Acceleration Platform (ACAP) designed for demanding AI and machine learning applications. It features two AI engines optimized for high-throughput and low-latency inference, alongside 1.5 million programmable logic cells, including 224K LUTs, providing design versatility. The board is equipped with four ARM Cortex-A72 cores and a single ARM Cortex-R5 real-time core, facilitating efficient computing and intelligent power management across various fields. It boasts 4GB of HBM2 for rapid memory access, 16GB of DDR4 SDRAM, and 32GB of integrated flash storage, complemented by PCIe Gen4 that offers a bandwidth of up to 64GB/s.The tech-

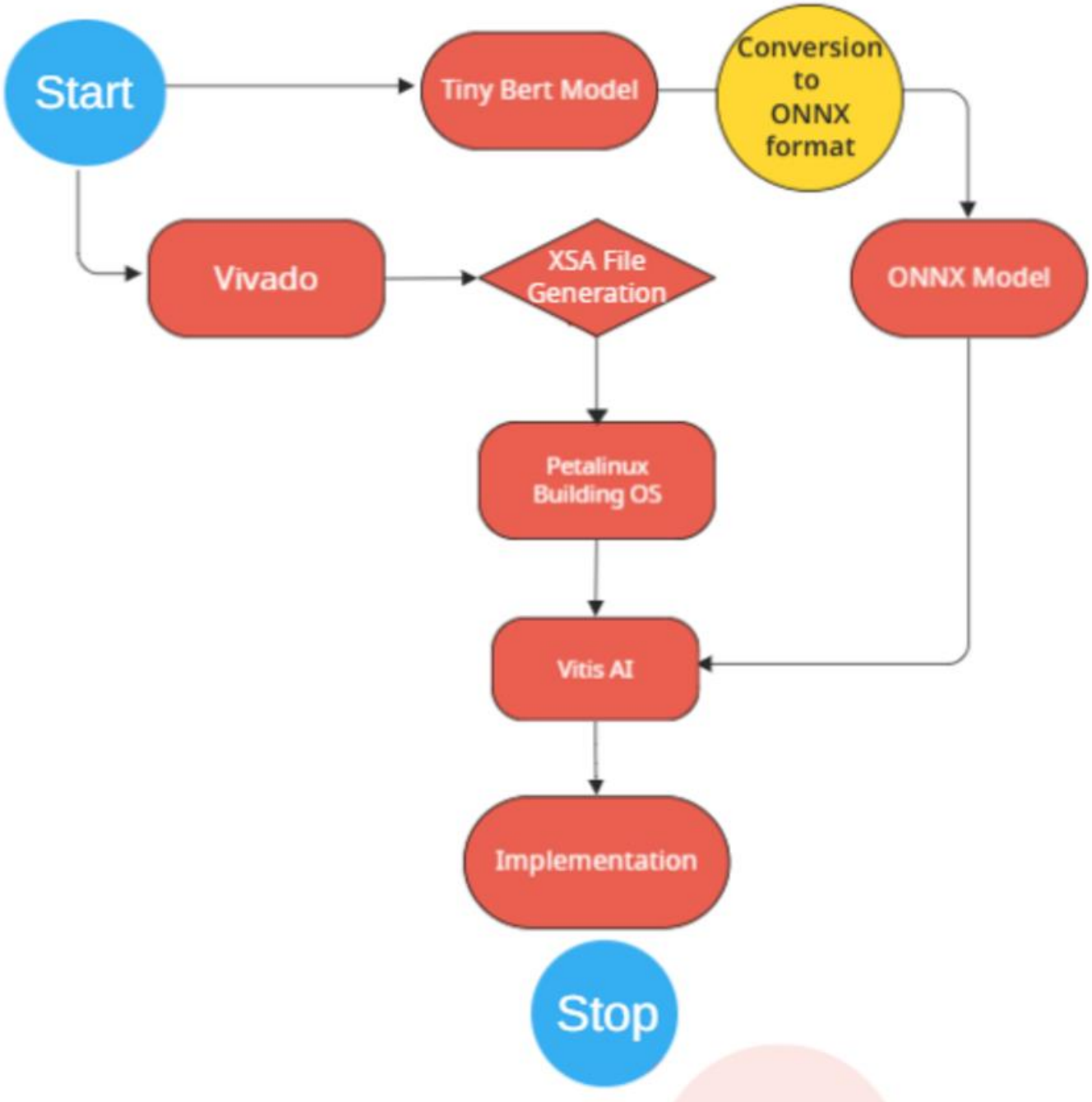
nology provides 2.5 TOPs with INT8 accuracy, making it perfect for AI workloads such as image recognition and real-time video processing, while consuming only 50-100W, substantially less than standard GPUs. Its energy economy and performance make it ideal for AI inference, autonomous systems, and industrial IoT applications like predictive maintenance and edge AI. .

COMPARISION BETWEEN EDGE COMPUTING and CLOUD COMPUTING:

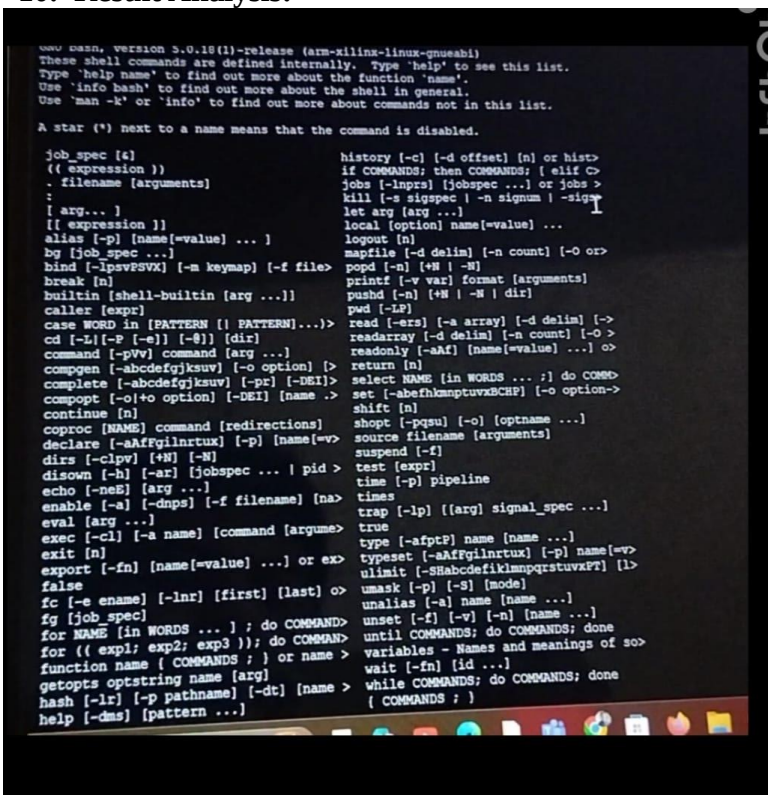
Aspect	Cloud Computing	Edge Computing
Definition	Centralized computing model that delivers resources over the internet.	Decentralized computing model that processes data close to its source.
Data Processing	Data is processed in remote data centers.	Data is processed locally, near the device or user.
Latency	Higher latency due to data traveling to remote servers.	Low latency as data is processed locally.
Use Cases	Web hosting, large-scale data analysis, AI model training.	Real-time processing like IoT, autonomous vehicles, and smart cities.
Scalability	Highly scalable with vast resources.	Limited scalability; depends on local hardware capabilities.
Dependency on Internet	Requires a stable internet connection for optimal performance.	Operates efficiently even with limited or no internet connection.
Bandwidth Usage	Consumes significant bandwidth for data transmission.	Reduces bandwidth usage by processing data locally.
Data Security	Data is stored in centralized servers, increasing risks of breaches if not secured.	Data stays local, reducing exposure to potential threats.
Cost Efficiency	Cost-effective for long-term and large-scale operations.	May involve higher upfront costs for hardware and maintenance.
Resource Allocation	Resources are allocated dynamically from a central pool.	Resources are fixed and depend on the edge devices' capacities.
Architecture	Centralized architecture with data centers in remote locations.	Distributed architecture with multiple edge nodes.
Real-time Capability	Limited real-time capabilities due to latency.	Excellent real-time capabilities.
Energy Efficiency	Energy-intensive due to centralized data processing and cooling systems.	Energy-efficient for specific tasks as it processes data quickly
Network Dependency	Dependent on robust and high-speed networks.	Less dependent on network stability.
Deployment Complexity	Easier to deploy with managed services from cloud providers.	Requires custom setup and management for edge devices.
Examples	AWS, Google Cloud, Microsoft Azure.	Smart homes, industrial IoT, autonomous vehicles.
Redundancy	High redundancy and failover mechanisms in cloud infrastructure.	Limited redundancy; failure of an edge device can disrupt operations.
Regulatory Compliance	Centralized compliance mechanisms managed by providers.	Easier to meet localized data regulations.
Adaptability to Changes	Highly adaptable to dynamic workloads.	Limited adaptability; scaling requires physical hardware upgrades.
Target Users	Enterprises requiring large-scale, centralized computing.	Applications needing localized, real-time processing and decision-making.

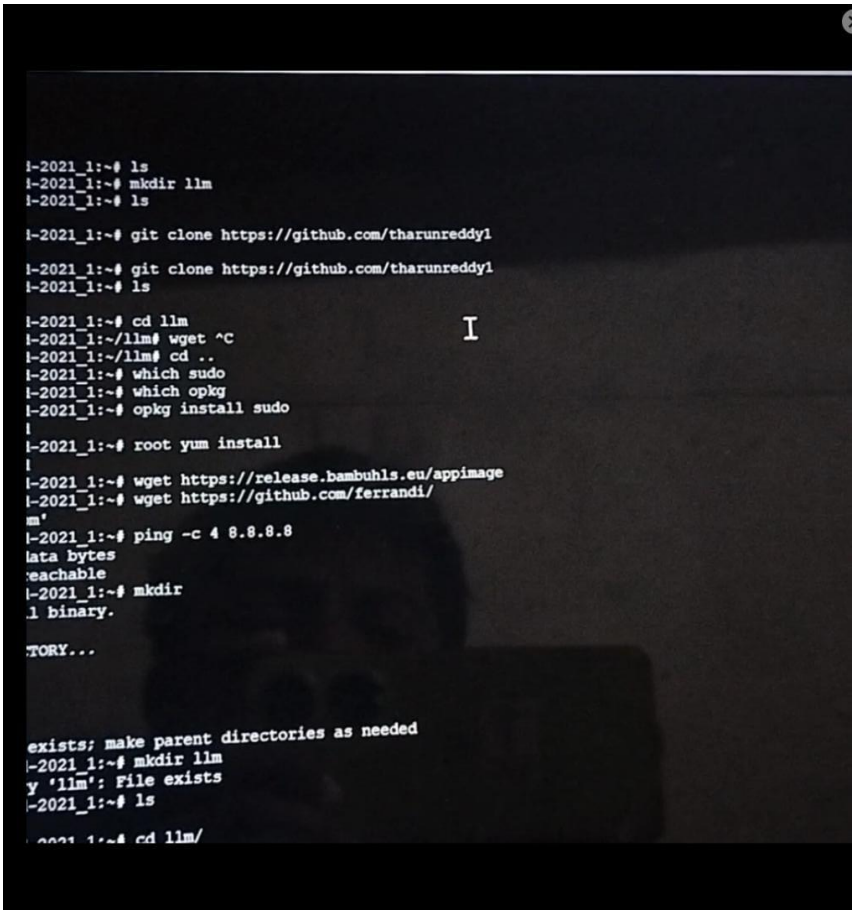
9. IMPLEMENTATION

Flow Chart:



10. Result Analysis:





11. FUTURE SCOPE:

This project can be expanded in the future such as quantum computing and neuromorphic computing. Furthermore, researching hybrid architectures—which integrate CPUs, GPUs, and specialized accelerators—may result in systems that are better suited for the implementation of LLM in which LLM has Billion of parameters. The project could be expanded to include edge/cloud computing environments, implementing customized LLMs, automated optimization methods, security and privacy issues, industry-specific deployment strategies, and an investigation of the moral and societal ramifications of implementing LLMs on various hardware platforms. The efficiency, scalability, and appropriate deployment of LLMs in a variety of applications are the goals of these future area.

12. CONCLUSION

In this study, we examined how well Large Language Models (LLMs) performed when implemented on FPGA. We set out to identify the best hardware platform based on energy efficiency, cost, and other considerations, in order to achieve efficient deployment.

By leveraging tools and methodologies to optimize partitioning and synthesis, we've demonstrated how performance, power efficiency, and scalability can be achieved in a range of applications. FPGA-based hardware-software codesign empowers developers to tailor solutions to specific needs, enhancing innovation in fields such as IoT, AI, and embedded systems. As technology advances, the integration of hardware and software will continue to drive cutting-edge developments, making it an exciting area to explore further.

Last Word of Advice:

We advise using high end FPGA-based solutions for installing LLMs due to the better performance and energy economy that they have shown, particularly in situations where power consumption and operating expenses are crucial factors. Due to their superior raw computing capacity, GPUs are still a great option for initial model training; however, FPGAs perform well in deployment situations, making them the best option for operating LLMs in production.

Businesses can realize substantial speed and efficiency gains by utilizing FPGAs, which will allow LLMs to reach their full potential across a range of applications. But we can say this advice based on estimation of FPGA Computational Power.

13. REFERENCES:

1. A Survey on Hardware Accelerators for Large Language Models (arxiv.org)
2. FlightLLM: Efficient Large Language Model Inference with a Complete Mapping Flow on different Hardware: 2401.03868 (arxiv.org)
3. https://arxiv.org/pdf/1706.03762.pdf
4. https://www.databricks.com/blog/us-air-force-hackathon-how-large-language-models-will-revolutionize-usaf-flight-test
5. https://www.achronix.com/blog/fpga-accelerated-large-language-models-used-chatgpt
6. https://www.aldec.com/en/company/blog/167–fpgas-vs-gpus-for-machine-learning-applications-which-one-is-better
7. To export a model : https://huggingface.co/docs/optimum/exporters/onnx/usageguides/export
8. transformers:https://huggingface.co/docs/transformers/serializationonnx
9. 2406.01698 (arxiv.org)
10. Han, S., et al. (2016). ”Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding.”
11. Shoeybi, M., et al. (2019). ”Megatron-LM: Training Multi-Billion Parameter Language Models Using Model Parallelism.”
12. https://www.cloudflare.com
13. https://www.techtarget.com/searchdatacenter/definition/edge-computing
14. 2309.13321.pdf Authors:Federico Manca,Francesco Ratto
15. petalinux-ug114.html(official Document)
16. Hardware-Software Codesign (Author: Tarek Darwish, Magdy Bayoumi)
17. Hardware/Software Codesign: The Past, the Present, and Predicting the Future(ieee paper)