IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

HARDWARE ACCELERATOR FOR SIGN LANGUAGE DETECTION ON FPGA

Shwetha V[1], Thushar Cherian[2], Varshith S[3], Prayag Singh[4], Vishalini Divakar[5]
Dept of ECE, K S Institute of Technology, Affiliated to VTU, Bengaluru, *India*[1-4]
Associate Professor, Dept of ECE, K S Institute of Technology, Affiliated to VTU, Bengaluru, *India*[5]

Abstract: This project is all about building a hardware accelerator for Convolutional Neural Networks (CNNs) on an FPGA, specifically designed for real-time sign language detection. By harnessing the parallel processing power of FPGA, the CNN model is implemented in Verilog, ensuring fast and efficient inference. To enhance performance, the trained model weights are seamlessly integrated into the hardware, enabling high-speed processing while keeping power consumption low. To validate the design, the implementation undergoes simulation, resource utilization analysis, and FPGA deployment, showcasing the immense potential of hardware accelerators for deep learning applications.

Index Terms - Hardware Accelerator, Convolution Neural Network, Field Programmable Gate Arrays, Sign Language Detection.

I. INTRODUCTION

With the growing demand for deep learning applications, Convolutional Neural Networks (CNNs) have become a fundamental tool in image processing, object recognition, and real-time decision-making. However, traditional computing platforms such as CPUs and GPUs often struggle with the computational demands of CNNs, leading to inefficiencies in speed and power consumption. To address these challenges, specialized hardware accelerators are gaining attention for their ability to improve processing efficiency while reducing energy usage.

This research focuses on developing a hardware accelerator for CNN inference, utilizing Field Programmable Gate Arrays (FPGAs) to achieve optimized performance. FPGAs offer significant advantages, including parallel computation, customizable architecture, and lower power consumption compared to conventional processors. By implementing CNN operations directly on FPGA hardware, this project aims to accelerate convolutional computations, minimize latency, and enhance throughput.

To demonstrate the effectiveness of this accelerator, we apply it to sign language detection as a practical use case. Sign language recognition is a crucial assistive technology that bridges the communication gap between individuals with hearing impairments and non-sign language users. By integrating CNN-based gesture recognition with FPGA-based acceleration, the system aims to achieve real-time, high-accuracy classification with minimal power consumption. This project not only highlights the potential of hardware acceleration for deep learning tasks but also contributes to making AI-driven communication tools more accessible and efficient.

II. LITERATURE SURVEY

[1] Convolutional Neural Networks Based Sign Language Interpreter: Varadala Sridhar's study focuses on translating American Sign Language (ASL) gestures into text using Convolutional Neural Networks (CNNs). Since ASL is widely used, automating its interpretation can improve communication for the deaf and mute community. The paper highlights advancements in gesture recognition through deep learning, which have significantly enhanced the accuracy of sign language recognition systems.

[2] Kurdish Sign Language Recognition Using CNN: This research presents a deep learning-based model for recognizing Kurdish Sign Language (KSL) gestures. The system aims to reduce reliance on human interpreters by automating alphabet recognition using CNNs. With a dataset of 132,000 hand images representing 33 symbols, the model achieves an impressive 99.87% accuracy by integrating CNNs with MediaPipe for real-time hand tracking, demonstrating the effectiveness of AI in sign language recognition.

[3] Accelerating Convolutional Neural Networks: FPGA-Based Architectures & Challenges: Haobo Ye's study explores the use of Field Programmable Gate Arrays (FPGAs) to accelerate CNNs, focusing on their energy efficiency and parallel computing capabilities. The paper introduces four FPGA-based CNN accelerator architectures, each designed for high-speed, low-power computations, and discusses various optimization strategies for improving CNN performance in real-time applications.

[4] Comparative Study of FPGA and GPU for High-Performance Computing & AI: This research compares FPGAs and GPUs in handling AI and high-performance computing (HPC) workloads. As computational demands rise, the paper evaluates the strengths of both architectures, emphasizing FPGAs for power-efficient, customizable processing and GPUs for high-speed parallelism, providing valuable insights into their respective use cases.

[5] FPGA Hardware Acceleration for Deep Learning: The study explores the role of FPGAs in accelerating CNNs for large-scale image processing. While CNNs excel in extracting features, their computational complexity often limits real-time processing. The paper highlights how FPGAs, with their reconfigurable nature and low power consumption, can effectively speed up CNN operations, making them ideal for deep learning applications requiring fast, stable, and efficient computing.

[6] Designing Deep Learning Hardware Accelerators & Efficiency Evaluation: This paper investigates hardware accelerators optimized for deep learning, particularly CNNs. As neural networks become more complex, traditional CPUs struggle to keep up with computational demands. The study emphasizes FPGA-based accelerators as an efficient alternative, leveraging high parallelism and lower power consumption to enhance performance while ensuring energy efficiency in real-time deep learning tasks.

[7] A Deep Learning Framework for Real-Time Sign Language Recognition Using Transfer Learning: This research introduces a deep learning-based system for real-time sign language recognition, utilizing transfer learning techniques to improve accuracy. The framework leverages deep learning models trained on extensive datasets to recognize gestures efficiently. To balance speed and accuracy, the system integrates TensorFlow object detection for enhanced recognition performance in real-time applications.

[8] Accelerating Neural Network Inference on FPGA-Based Platforms – A Survey: This study explores neural network acceleration on FPGA-based platforms, emphasizing their advantage over traditional CPUs and GPUs in power efficiency and performance. The paper discusses various optimization strategies, including hardware/software co-design, parallel processing, and efficient data management, which are critical for enhancing deep learning inference speed on FPGA hardware.

[9] Sign Language Recognition: A Deep Survey: This paper provides a comprehensive review of sign language recognition (SLR) techniques, categorizing them into vision-based and sensor-based approaches. It highlights the growing reliance on CNNs and RNNs for improving sign language translation accuracy while also acknowledging challenges such as variability in signing styles, occlusions, and dataset limitations, which can impact real-world performance.

[10] High-Throughput & Power-Efficient FPGA Implementation of YOLO CNN for Object Detection: This study introduces an FPGA-based YOLO object detection system, optimized for speed and energy efficiency. Traditional CNNs suffer from high latency due to frequent external memory accesses, leading to increased power consumption. To address this, the research proposes quantizing YOLO CNN with binary weights, storing the network model in FPGA block RAMs to minimize off-chip memory usage. Additionally, pipelined convolutional layers ensure continuous data flow and improved computational efficiency, making the system well-suited for real-time object detection tasks.

III. METHODOLOGY

1. Block Diagram

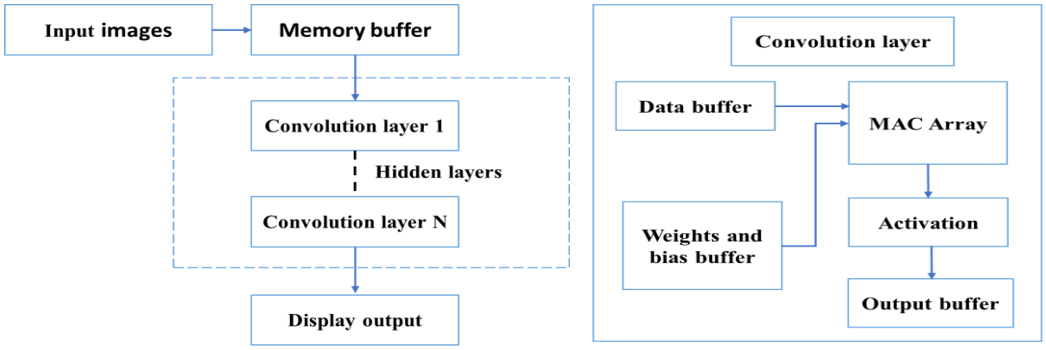

Figure 1. Block Diagram

The FPGA-based CNN accelerator is specifically designed to recognize sign language gestures efficiently by utilizing parallel processing for faster computations. This architecture integrates multiple components that work together to ensure accurate and smooth gesture classification in real time. The process begins by capturing sign language images from a dataset, which then go through preprocessing steps like noise reduction, contrast adjustment, and resizing to enhance image quality for FPGA processing. A memory buffer temporarily stores these processed images, ensuring seamless data transfer between storage and computational units. At the core of the system is the convolutional processing unit, which plays a crucial role in extracting key features from the input images.

This unit consists of several submodules, including a Multiply-Accumulate (MAC) unit for performing convolution operations, ReLU activation to introduce non-linearity, and dedicated memory for storing weights and biases. Once features are extracted, pooling layers refine the data, and the fully connected layers classify the gestures based on learned patterns. After classification, the system instantly displays the recognized gesture in real-time. The entire setup is optimized for the Altera DE10-Lite FPGA, enabling high-speed, low-power inference.

2. CNN Model Training

The first step in this project involves designing and training a Convolutional Neural Network (CNN) using TensorFlow to recognize sign language gestures. The model learns from a labeled dataset of gesture images, gradually refining its ability to differentiate between various signs through multiple convolutional layers. To improve efficiency, a convolutional layer with a stride of 4 is used instead of traditional pooling layers. This adjustment reduces computational complexity and parameter count while still preserving the key features required for accurate classification. The model is inspired by LeNet-5, but modified specifically for sign language recognition, making it lightweight and well-suited for real-time applications without sacrificing performance.

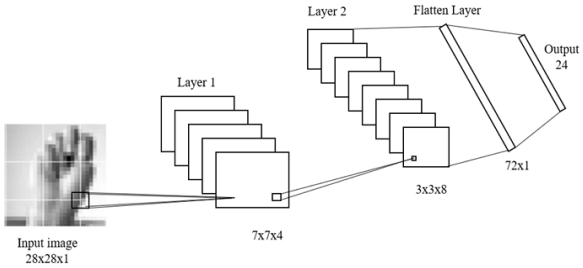

Figure 2. Modified Lenet-5 CNN architecture architecture with their parameters

Model: "sequential_1"

Layer (type)	Output Shape	Param #
conv2d_2 (Conv2D)	(None, 7, 7, 4)	68
conv2d_3 (Conv2D)	(None, 3, 3, 8)	296
flatten_1 (Flatten)	(None, 72)	0
dense_1 (Dense)	(None, 24)	1,752

Total params: 2,116 (8.27 KB)
Trainable params: 2,116 (8.27 KB)
Non-trainable params: 0 (0.00 B)

Figure 3. Different layers of the implemented CNN

The CNN model has been rigorously trained on a carefully curated dataset, achieving an impressive 97.6% accuracy (as shown in Figure 3). Such a high success rate highlights the model's ability to accurately classify a broad range of gestures with minimal errors. The training process involved fine-tuning network parameters, optimizing convolutional filters, and adjusting weights, ensuring better feature extraction. This level of optimization enables the system to detect intricate hand movements and subtle variations in gestures, significantly improving recognition accuracy.

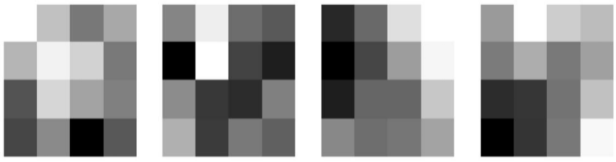

Figure 4. Weights of layer-1 of the implemented CNN model

Since FPGAs do not natively support floating-point operations, the trained model's weights undergo conversion into a fixed-point format to enhance memory efficiency and computational precision. This transformation is crucial for real-time processing while maintaining high accuracy. Once converted, these weights are mapped to FPGA memory buffers, ensuring fast convolution operations without compromising performance. By adapting the model for FPGA deployment, this approach offers a power-efficient, scalable, and optimized deep learning accelerator tailored for real-world sign language detection.

3. FPGA Implementation

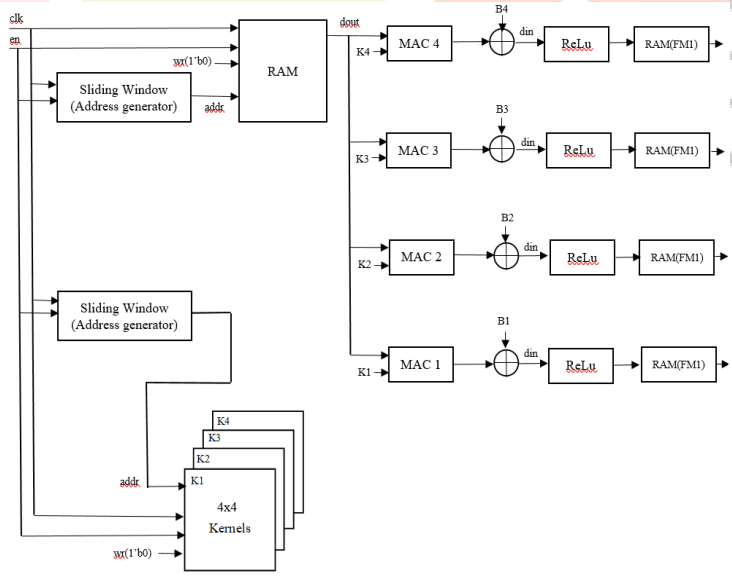

Figure 6. Block Diagram of Layer-1 of CNN architecture

The first layer of this FPGA-based CNN is optimized for high-speed computation using parallel Multiply-Accumulate (MAC) units. Input feature maps are stored in RAM, and a Sliding Window Address Generator extracts 4×4 patches for convolution. The FPGA-based CNN accelerator enhances performance by using multiple MAC units to process feature map sections in parallel, ensuring fast and efficient computation. A Sliding Window Address Generator scans the 7×7 feature maps, extracting 3×3 sections for convolution with pre-trained filters in the MAC block. Bias values are added post-convolution, followed by ReLU activation to eliminate negative values.

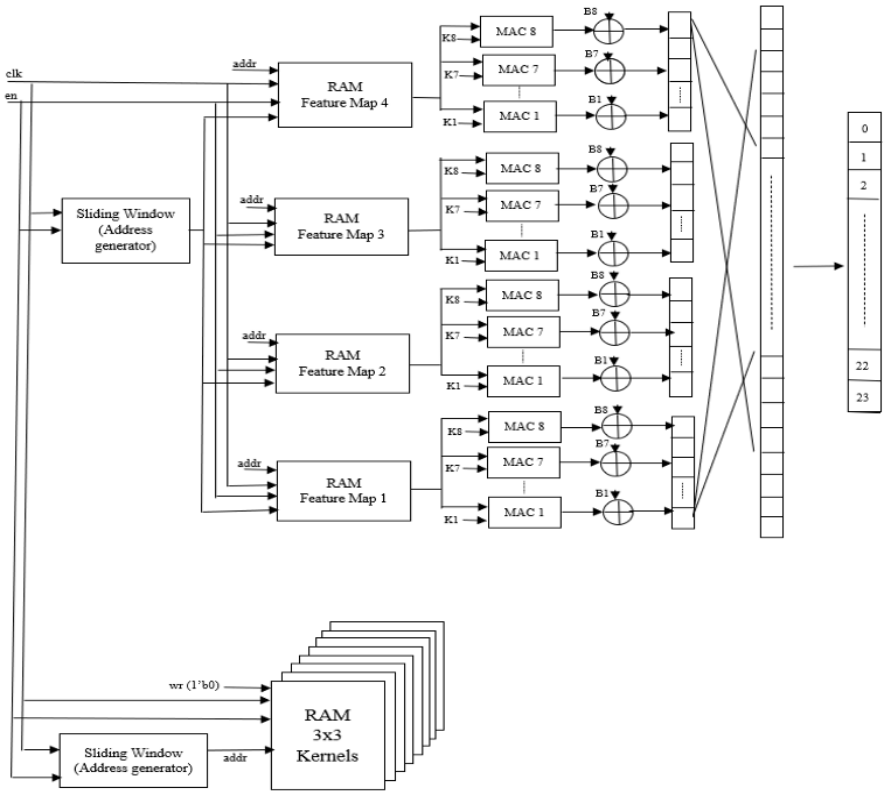

Figure 7. Block Diagram of Layer-2, flatten and output layer.

This FPGA-based implementation enables real-time, low-latency sign language detection with exceptional accuracy and energy efficiency. The structured pipeline ensures that each CNN layer seamlessly processes gestures, delivering a high-performance AI solution capable of operating in real-world assistive applications.

IV. RESULTS

1. LAYER-1 IMPLEMENTATION

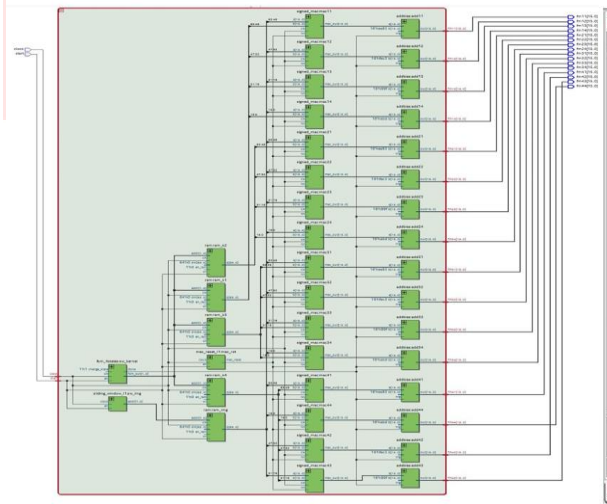

Figure 9. Layer-1 RTL netlist.

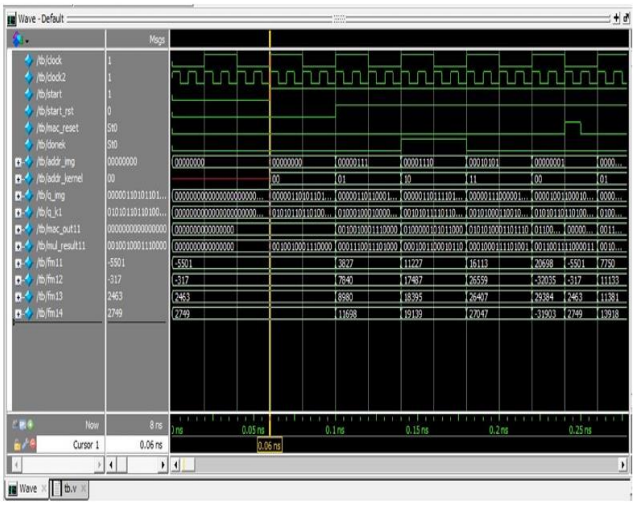

Figure 10. Layer-1 Simulation output

The RTL netlist for Layer 1, as shown in figure.9., generated after synthesizing the Verilog code in Quartus Prime, offers a gate-level view of the design optimized for FPGA implementation. It highlights the connections between core components like the Sliding Window Generator, MAC unit, Activation Function, and Memory Buffers. The simulation of Layer 1 of the CNN architecture was conducted to validate its functional correctness and performance before hardware implementation. The testbench was designed to apply multiple test cases, ensuring that the convolution operation, activation function, and memory interactions were accurately executed.

2. CNN HARDWARE IMPLEMENTATION

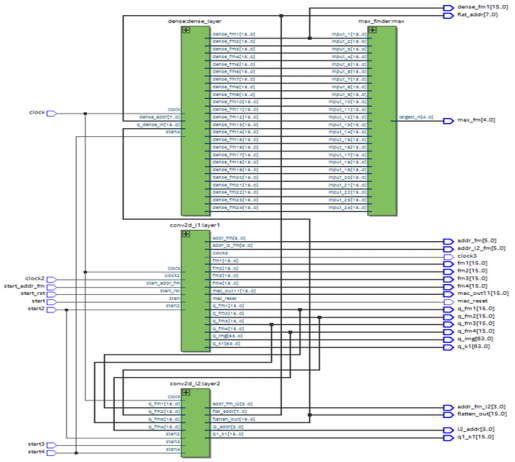

Figure 11. RTL netlist of CNN implementation

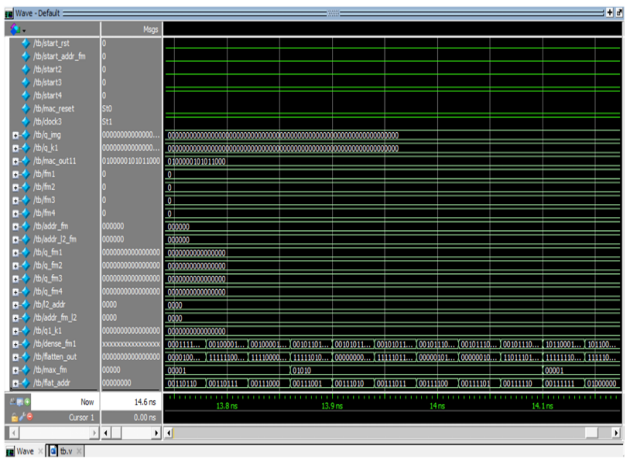

Figure 12. Simulation output of CNN Implementation

After verifying Layer 1, the full CNN was developed in Verilog and simulated to assess its performance on FPGA for sign language detection. The design integrates multiple convolutional and activation layers along with a fully connected layer to handle image inputs effectively. Using Quartus Prime, the RTL netlist was generated to visualize the gate-level structure. This helped evaluate data flow, resource usage, and pipeline efficiency, confirming that key elements like convolutions, multipliers, and memory units were efficiently mapped onto the FPGA. The functional simulation was obtained using ModelSim, where the CNN was tested with multiple input images to ensure correct feature extraction and classification. The simulated outputs were compared with the results from the Python-based CNN model, verifying that the FPGA implementation maintained high accuracy while operating with fixed-point arithmetic. The simulation waveforms demonstrated that the outputs at each stage—convolution, activation, and classification—were correctly processed, confirming the validity of the hardware design.

3. RESOURCE UTILIZATION AND POWER ANALYZER REPORT

```
Fitter Status : Successful - Mon Mar 17 10:42:26 2025
Quartus Prime Version : 23.1std.1 Build 993 05/14/2024 SC Lite Edition
Revision Name : ha_cnn_top
Top-level Entity Name : ha_cnn_top
Family : MAX 10
Device : 10M50DAF484C7G
Timing Models : Final
Total logic elements : 2,446 / 49,760 ( 5 % )
    Total combinational functions : 2,446 / 49,760 ( 5 % )
    Dedicated logic registers : 168 / 49,760 ( < 1 % )
Total registers : 168
Total pins : 16 / 360 ( 4 % )
Total virtual pins : 0
Total memory bits : 0 / 1,677,312 ( 0 % )
Embedded Multiplier 9-bit elements : 20 / 288 ( 7 % )
Total PLLs : 0 / 4 ( 0 % )
UFM blocks : 0 / 1 ( 0 % )
ADC blocks : 0 / 2 ( 0 % )
```

Figure 13. Resource Utilization

```
Power Analyzer Status : Successful - Mon Mar 17 10:53:11 2025
Quartus Prime Version : 23.1std.1 Build 993 05/14/2024 SC Lite Edition
Revision Name : ha_cnn_top
Top-level Entity Name : ha_cnn_top
Family : MAX 10
Device : 10M50DAF484C7G
Power Models : Final
Total Thermal Power Dissipation : 98.48 mW
Core Dynamic Thermal Power Dissipation : 0.00 mW
Core Static Thermal Power Dissipation : 89.94 mW
I/O Thermal Power Dissipation : 8.54 mW
Power Estimation Confidence : Low: user provided insufficient toggle rate data
```

Figure 14. Power Analyzer report

The Quartus Prime fitter report for the ha_cnn_top design on Intel MAX 10 (10M50DAF484C7G) confirms successful compilation, utilizing 5% of logic elements (2,446/49,760), 7% of embedded multipliers (20/288), and 4% of available pins (16/360). The design primarily consists of combinational functions with minimal register usage (168 registers). No memory bits, PLLs, UFM blocks, or ADC blocks are utilized, indicating potential for further optimization or resource allocation adjustments. The power analysis for the ha_cnn_top design on an Intel MAX 10 (10M50DAF484C7G) FPGA was successfully performed using Quartus Prime 23.1. The total thermal power dissipation is 98.48 mW, with 89.94 mW attributed to core static thermal dissipation and 8.54 mW to I/O thermal dissipation. The core dynamic thermal power dissipation is 0.00 mW, indicating minimal dynamic switching activity.

Figure 15. Output on FPGA board

Parameters	Implementation of Lenet-5 on FPGA [1]	Proposed Work in This Paper
FPGA Kit Used	Kintex-7	Altera DE-10 Lite
Logic Utilization	14659/53200 (27.55%)	2446/49760 (5%)
DSP Blocks	125/220 (56.82%)	20/288 (7%)
Model Accuracy	Not Mentioned	97.09%
Inference Time	19.1 µs	16 µs
Power Consumption	Not Mentioned	98.48 mW

Table I. Comparison of results of proposed work with previously published work

The FPGA-based CNN accelerator successfully displayed the final output on the board, processing pre-stored images with efficient hardware acceleration. Running at 20 MHz, it completed inference in just **16 µs** over 320 clock cycles, demonstrating impressive speed. A comparative analysis follows, evaluating inference time, power efficiency, resource utilization, and accuracy against existing research. The Table I, clearly shows that the proposed work achieves better power efficiency, lower inference time, and significantly lower resource utilization while maintaining high model accuracy.

V. CONCLUSION

This project revolutionizes deep learning acceleration by leveraging FPGA technology for real-time CNN processing. Unlike power-hungry CPUs and GPUs, it maximizes efficiency through parallelism and pipelining. Using Q4.12 fixed-point arithmetic, it balances accuracy and performance, enabling fast, low-power sign language recognition and paving the way for embedded AI advancements.

VI. APPLICATIONS

HARDWARE ACCELERATOR FOR CNN:

1. Autonomous Vehicles: Real-time image and object recognition for safe navigation.
2. Healthcare Diagnostics: High-speed medical image analysis, such as tumor detection or disease classification.
3. Industrial Automation: Visual inspection and defect detection in manufacturing.
4. Smart Surveillance: Real-time facial recognition and anomaly detection in security systems.
5. Robotics: Enabling robots to process vision tasks efficiently for autonomous operation

SIGN LANGUAGE DETECTION:

1. Assistive Communication Devices: Portable systems for real-time translation of sign language into text or speech.

VII. FUTURE SCOPE

1. Integrating real-time camera input
2. Multimodal Recognition
3. Real-World Deployment & Testing

VIII. REFERENCES

[1]Varadala Sridhar - Convolutional Neural Networks Based Sign Language Interpreter; International Journal of All Research Education and Scientific Methods (IJARESM),ISSN: 2455-6211, Volume 12, Issue 9, September-(2024)

[2] Sarkhel H. Taher Karim, Muhammed Latif Mahmood, Siva Sabir Abdulla, Shano Ali Abdulla Kurdish - Sign Language Recognition Using Convolutional Neural Network; Journal of Telecommunication, Electronic and Computer Engineering ISSN: 2180 – 1843 e-ISSN: 2289-8131 Vol. 16 No. 3 (2024)

[3] Haobo Ye - Accelerating Convolutional Neural Networks: Exploring Fpga-based Architectures And Challenges; CONF-MSS 2024 Journal of Physics: Conference Series 2786 (2024) 012004

[4] Muthukumaran Vaithianathan, Mahesh Patil, Shunyee Frank Ng, Shiv Udkar - Comparative Study Of Fpga And Gpu For High-performance Computing And Ai; ESP International Journal of Advancements in Computational Technology ISSN: 2583-8628 / Volume 1 Issue 1 May 2023 / Page No: 37-46 (2023)

[5] Haochen Shi – FPGA Hardware Acceleration Design for Deep Learning; Highlights in Science, Engineering and Technology CMLAI 2023 Volume 39 (2023)

[6] Zhi Qi, Weijian Chan, Rizwan Ali Naqvi, Kamran Siddique – Designing Deep Learning Hardware Accelerator And Efficiency Evaluation; Hindawi Computational Intelligence and Neuroscience Volume 2022, Article ID 1291103, 11 pages (2022)

[7] Vijeeta Patil, Sujatha C, Shridhar Allagi, Balachandra Chikkoppa – A Deep Learning Framework For Real time Sign Language Recognition Based On Transfer Learning; International Journal of Engineering Trends and Technology Volume 70 Issue 6, 32-41, June 2022 ISSN: 2231 – 5381(2022)

[8] Ran Wu, Xinmin Guo, Jian Du, Junbao Li – Accelerating Neural Network Inference On Fpga-based Platforms—a Survey; Electronics 2021, 10, 1025. https://doi.org/10.3390/electronics10091025 (2021)

9] Razieh Rastgoo, Kourosh Kiani, Sergio Escalera – Sign Language Recognition: A Deep Survey; Expert Systems with Applications Volume 164, February 2021, 113794 (2021)

[10] Duy Thanh Nguyen, Tuan Nghia Nguyen, Hyun Kim, Hyuk-Jae Lee – A High-throughput And Power efficient FPGA Implementation Of Yolo CNN For Object Detection; IEEE Transactions on Very Large Scale Integration (VLSI) Systems (Volume: 27, Issue: 8, August (2019))

[11] https://www.kaggle.com/datasets/datamunge/sign-language-mnist