Journal of Information Systems Engineering and Management
2025, 10(31s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Verilog based ANN For Handwritten Alphabet Recognition using a Fully Connected Architecture

N Neelima[1], N. Lakshman Pratap[2], Aruru Sai Kumar[3], Janjirala Rohan[4], K Yashwanth[5], S Sai Madhava Reddy[6]
[1,3,4,5,6]*Department of ECE, VNR Vignana Jyothi Institute of Engineering and Technology, Telangana, India*
[2]*Department of ECE, Malla Reddy Engineering College, Telangana, India.*
Email: neelima_n@vnrvjiet.in[1], n.l.pratap@gmail.com[2], asaikumar.nitw@gmail.com[3]

ARTICLE INFO

Received: 30 Dec 2024

Revised: 12 Feb 2025

Accepted: 26 Feb 2025

ABSTRACT

Manufactured Insights (AI) and Machine Learning (ML) are increasingly becoming apparatus usage, mainly in neural network models. This extension offers a Verilog-based Manufactured Neural Network (ANN) designed for handwritten letter set recognition on an FPGA. The ANN is a 5-layer fully connected technology-based architecture, making use of ReLU and Sigmoid activation functions for efficient pattern recognition Usage is optimized with respect to resource consumption and control use, ensuring that it can work in real time. The plan employs FPGA parallelism to boost the performance speed, but with the minimal usage of DSP and memory. Comparisons with conventional floating-point ANN models indicate improvements achieved through fixed-point arithmetic and binarized neural systems (BNNs), which reduce complexity in computation. Additionally, successful experiences from the Perceptron calculations and VHDL-based ANN models are directions for improvement in weight capacity and enactment work executions. The organize is prepared utilizing the MNIST dataset, accomplishing 98% accuracy, with equipment testing and confirmation conducted in Vivado. The venture illustrates the potential of FPGA-based ANNs for real-time, low-power applications in character acknowledgment and advanced flag handling. Future advancements incorporate half breed ANN-perceptron models and encourage optimization of weight accuracy for upgraded proficiency. This work contributes to the developing field of hardware-accelerated neural systems, advertising a adaptable and productive approach to FPGA-based AI.

Keywords: *Artificial Neural Networks, Handwritten Recognition, Resource Optimization, Neurons, Mathematical Model*

I. INTRODUCTION

Handwritten character recognition is a crucial challenge in machine learning and digital signal processing, with applications spanning optical character recognition (OCR), human-machine interaction, and automated documentation systems [1-4]. Traditional software-based solutions for character recognition, while effective, often struggle to meet the real-time performance requirements of embedded and hardware-constrained environments. This limitation arises due to the high computational complexity of neural networks and the inability of general-purpose processors to execute them efficiently in low-power, high-speed applications. To overcome these challenges, hardware implementations of neural networks, particularly on Field Programmable Gate Arrays (FPGAs), provide a compelling alternative. FPGAs offer parallel processing, high-speed computation, and reconfigurability, making them ideal for accelerating deep learning tasks in real-time. Unlike CPUs and GPUs, which rely on sequential execution or power-hungry architectures, FPGAs can execute multiple neural computations in parallel, leading to significant improvements in speed and energy efficiency [5-8].

Designing and implementing a fully connected artificial neural network (ANN) for handwritten character recognition requires a robust hardware description language (HDL). Verilog serves as an ideal choice due to its capability to model, simulate, and synthesize complex digital circuits. This project leverages Verilog to construct an ANN that can be directly deployed on FPGA platforms, ensuring efficient hardware-based computation. The use of FPGA-based implementations enhances processing speed, power efficiency, and parallelism compared to traditional software-based approaches. Handwritten character recognition is a challenging task due to the variations in writing styles, noise, and distortions. By employing an ANN, the system can learn intricate patterns and classify characters with high accuracy. The fully connected nature of the ANN ensures that each neuron in one layer is connected to every neuron in the next, facilitating comprehensive feature extraction. This interconnected structure allows the network to generalize well across diverse handwriting samples[9-11]. Thus, integrating ANN with Verilog-based FPGA design provides a powerful solution for real-time character recognition.

A fully connected ANN architecture plays a crucial role in enhancing recognition accuracy by capturing complex relationships within the input data. Each neuron processes information from multiple sources, allowing the network to identify subtle variations in handwritten characters. The ANN consists of multiple layers, including an input layer, one or more hidden layers, and an output layer. The hidden layers contain activation functions that introduce non-linearity, enabling the network to model intricate data distributions. The training phase involves adjusting connection weights using optimization techniques such as backpropagation. Once trained, the ANN can classify new handwritten characters based on learned patterns. Implementing this ANN in Verilog ensures that the hardware design supports parallel computations, significantly improving processing speed. The FPGA-based realization of the network enables real-time recognition, making it suitable for embedded applications [12-14]. Unlike software-based approaches, hardware implementations provide deterministic performance with minimal latency. This makes Verilog an optimal choice for designing high-performance ANN architectures.

The integration of ANN with FPGA platforms offers several advantages, including scalability, flexibility, and high-speed processing. Verilog allows precise control over hardware resources, optimizing power consumption and computational efficiency. Additionally, FPGA-based implementations can be reconfigured to accommodate different network architectures or upgraded to support new character sets. This adaptability is essential for practical applications such as postal sorting, bank check verification, and digital handwriting recognition. The modular nature of Verilog facilitates the design and testing of individual components, ensuring reliability before full-scale deployment. Hardware-level parallelism in FPGA enables simultaneous execution of multiple neurons, significantly reducing inference time. Moreover, the compact nature of FPGA-based systems makes them suitable for portable and embedded applications [15-17]. The ability to implement ANN architectures in hardware bridges the gap between software-based deep learning and real-time processing demands. As a result, this project demonstrates an efficient approach to character recognition using ANN and Verilog.

II. MATHEMATICAL FORMULATION

The ANN operations can be expressed as:

$$y_i = f\left(\sum_{j=1}^{n} w_{ij}x_j + b_i\right)$$

Where: - w_{ij} : weight connecting input j to neuron i, - x_j : input values, - b_i: bias term, - f(x): activation function.

The activation functions used are:

$$f(x) = \begin{cases} x, & if\ x > 0\ (ReLU) \\ \dfrac{1}{1+e^{-x}} & (sigmoid) \end{cases}$$

For classification, the output yi is determined by the softmax function:

$$y_i = \frac{e^{z_i}}{\sum_{k=1}^{n} e^{z_k}}$$

The network minimizes the cross-entropy loss:

$$L = -\frac{1}{N}\sum_{i=1}^{N}[t_i \log(y_i) + (1-t_i)\log(1-y_i)]$$

These equations form the theoretical basis of the network implemented in Verilog.

III. LITERATURE REVIEW

The valuable insights in training ANNs with FPGAs, especially on IEEE 754 floating-point operations and optimizing resources such as expansion and sigmoid activation for hardware. By shifting ANN operations from the program to the hardware, it seems how FPGA-based structures can exploit parallelism for efficient training and induction, thus making it worthwhile for high-speed applications. These techniques may advance neural net projects by enhancing the speed and efficiency of trainability [18].

The knowledge into the actualization of ANN training on FPGAs by focusing on IEEE 754 floating-point operations and optimizing capacities like expansion and sigmoid enactment for hardware. The movement of ANN operations from computer program to equipment unravels how FPGA-based structures can misuse parallelism to come up with proficient preparing and induction, making them fundamentally necessary for high-speed applications. Such strategies seem to move forward neural arrange ventures by improving both preparing speed and proficiency [19].

Simple knowledge in relation to the neural network models by describing the McCulloch-Pitts model, which uses a variable synaptic weight and a summing device to model neuron activity. Its focus on threshold-based dual outputs provides a relatively simplified basis for implementing fully connected neural models in Verilog. This helps extend by encouraging the development of an integrated framework for designing basic neural operations in hardware for tasks such as recognizing a set of written letters[20].

The reference explores the founding fathers in the artificial neural networks (ANNs) and deep learning and specifically Warren McCulloch, Walter Pitts, and Frank Rosenblatt, who are pioneers of the neural models and structures that have been used in modern computer systems. Their breakthrough led to double neural training as well as developing what would later be called the perceptron - a precursor to sophisticated multilayer systems. This legacy influences ANN-based models such as those utilized in ”VERILOG Based ANN for Transcribed Letter set Acknowledgment Employing a Completely Associated Design,” wherein such standards are applied in computerized design acknowledgment assignments[21].

The reasoning on arithmetic and reasoning within ML as supporting areas such as information processing, classification, and neural networks. All these are making up the core elements of ANN models, namely Manufactured Neural Systems. This knowledge empowers ventures like ”VERILOG Based ANN for Written by hand Letter set Acknowledgment Employing a Completely Associated Design” in providing the scientific standards that give direction toward calculating formulas for design acknowledgment tasks. Slope plunge and relapse, among others, have been underscored as critical for building influential and accurate models [22].

The effective FPGA-based ANN demonstration for written by- hand digit recognition, which achieves 98% accuracy with minimal control and memory usage, making it comparable to GPU-based schemes in terms of speed as well as resource usage. This led the “VERILOG Based ANN for Manually written Letter set Acknowledgment” extension to present insights on the points of interest of fully connected FPGA structures compared to GPUs, especially in reference to control efficacy and idleness. Operations such as fixed-point representation and sigmoid enactment were a fundamental basis for enhancing the FPGA’s performance in comparable manually written acknowledgment tasks [23].

The utilization of fixed-point arithmetic has become an essential methodology for neural networks implemented on FPGA, facilitating effective resource management while preserving precision. Trainable quantization approaches, including QFX [7], adaptively modify the binary-point location throughout the training process to enhance DSP efficiency and reduce hardware requirements. Furthermore, QFX incorporates a multiplier-free quantization approach, which markedly decreases both computational complexity and energy usage, rendering it appropriate for FPGAs with limited resources. Compared to floating-point arithmetic, fixed-point designs realize considerable resource savings, which amounts to up to 70% fewer LUTs and DSPs with almost negligible loss in accuracy. Additionally, quantization-aware training, as demonstrated in [24], can reduce the loss in accuracy due to the adaptation of the model during training. These techniques have been applied very successfully to handwritten recognition, among other applications, where FPGA-based implementations achieve low power, high-speed processing capabilities [24].

The implementing of 5-layer fully connected ANN in Verilog for handwritten character recognition on an FPGA. This paper explores Binarized Neural Networks (BNNs), which can fully utilize your FPGA resource. The current ANN usage employs ReLU and Sigmoid activations, which entail increments and memory to store floating-point weights. BNNs supplant weights and actuations with parallel values (+1 or -1), killing multipliers and decreasing control utilization. This will diminish LUT utilization, DSP piece utilization, and memory impression, as appeared in Vivado blend report. Rather than complex floating-point operations, BNNs utilize bitwise operations (XOR, AND), which are much quicker in FPGA equipment. This will offer assistance progress the speed and effectiveness of your ANN, making it more appropriate for real-time character acknowledgment. ANN as of now accomplishes 98 accuracy, whereas BNNs might somewhat decrease exactness but incredibly improve control effectiveness and use crossover layers, where early layers utilize BNN-based preparing, whereas afterward layers utilize higher accuracy actuations for made strides precision. Replace floating-point weights with double values (store efficiently in LUTs or shift registers). Optimize increment operations to use simple bitwise XOR operations and memory access designs in Verilog by using parallel weight capacity[25].

IV. METHODOLOGY

It is necessary to create a method called "Verilog-based Artificial Neural Network (ANN) for Handwritten Alphabet Recognition Using a Fully Connected Architecture" that combines neural network architecture, hardware design, and handwriting identification—especially in Verilog. Here's how to approach this task methodically:

1.Problem Statement:

Create an ANN in Verilog to recognize written by hand letter set letters employing a completely associated neural arrange design. Input/Output: The input would be an picture or pixel information of manually written letters (A-Z), and the yield ought to be the anticipated letter set character.

2. Neural Network Structure:

It is basically a design of three layers they are :

a.Input Layer Input to the ANN will be a matrix of pixel values.The size of the input layer corresponds to the number of pixels in the image (e.g. , for 28x28 images, the input layer will have 784 nodes).

b.The number of hidden layers and neurons per layer depends on the complexity of the task and available hardware resources. For instance, two hidden layers with 128 and 64 neurons each could be a good balance for a simple handwriting recognition task.

c. Output Layer The output layer will have 26 neurons (one for each letter from 'A' to 'Z').

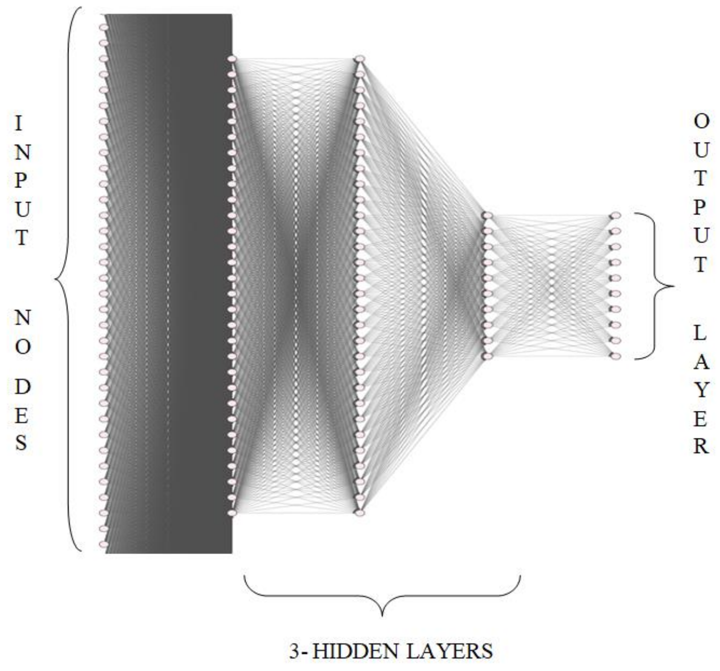

Fig. 1. Architecture of Neural Network.

3.Activation Function:

We have used ReLU and sigmoid functions in the hidden layers For example, ReLU can be approximated by simple conditional logic (if greater than 0, pass the value; otherwise, output 0).The final layer of an ANN for handwritten recognition typically uses Softmax .Converts raw scores (logits) into probabilities.Helps in multi-class classification (e.g., A-Z recognition).Hardware alternative: Use Argmax (finds the neuron with the highest activation).

ReLU, Sigmoid, and Softmax are commonly used activation functions in neural networks, each serving a specific purpose. ReLU (Rectified Linear Unit) is computationally efficient and widely used in hidden layers due to its simple operation—outputting the input if positive and zero otherwise. This makes it ideal for deep networks and FPGA-based implementations, though it suffers from the dying ReLU problem, where neurons can become inactive for negative inputs. Sigmoid squashes values into the range (0,1), making it useful for binary classification, but it has vanishing gradient issues, which slow down learning in deep networks. Softmax, on the other hand, is used in the output layer for multi-class classification problems, such as handwritten alphabet recognition, as it converts logits into probabilities that sum to 1. While Sigmoid and Softmax require exponentiation and division, making them computationally expensive, ReLU is the best choice for hidden layers in FPGA-based implementations due to its simplicity and efficiency.

4.Training of the Neural Network:

a.The training the data in python using a library called "TensorFlow" and "PyTorch".

b.We used the MNIST dataset which contains our handwritten alphabets.

c.The process is that the dataset to a settled measure (e.g., 28x28 pixels). Normalize pixel values (e.g., between and 1). Prepare the ANN demonstrate on this dataset until accomplishing a great precision i.e., 98%. Save the model weights and architecture.

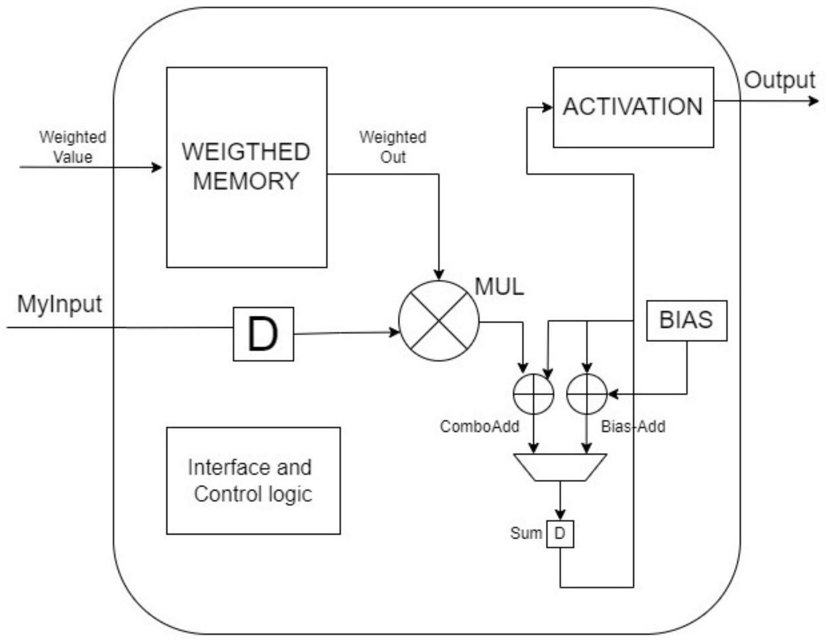

Fig. 2. Block diagram of Neuron.

5.Implementation in Verilog:

Each neuron within the ANN can be executed as a little computational unit in Verilog, performing the weighted entirety of inputs taken after by an enactment work.

a. Input Layer Handling: For each pixel within the picture,multiplier circuit will increase the pixel esteem with the comparing weights and adders will entirety the duplicated values.

Our implementation uses fixed-point arithmetic for efficiency in computation, motivated by techniques such as QFX [7], which optimize the position of the binary point during training of the model to minimize resource usage while maintaining accuracy.

b. Covered up Layer Preparing: Utilize comparable multiplieradder structures for each neuron. Execute the actuation work as combinational logic.

c. Yield Layer: Prepare the ultimate layer so also, with 26 yields speaking to each letter set character. Apply a thresholding component to decide the recognized letter based on the most noteworthy likelihood yield.

d. Control Logic in Network: Actualize control rationale to oversee the feed-forward operation of the arrange. The clock flag will decide when the neuron yields are computed, passed on to the following layer, and at last yield.

6.Testing and Verification:

Testbench Improvement: Compose a Verilog testbench to reenact the organize. Utilize a subset of the dataset for approval.

Input Simulation: Give pixel values as double or fixed-point inputs to the Verilog demonstrate. Yield verification: Compare the Verilog model's yield with anticipated comes about to approve its exactness.

Error Analysis: Check and record occasions where the demonstrate comes up short to recognize digits, analyze exactness, and make alterations on the off chance that required.

V. IMPLEMENTATION DETAILS OF THE NEURON MODULE

The neuron module serves as the core computational unit in the artificial neural network (ANN), performing essential operations such as processing multiple inputs, computing weighted sums, and applying an activation function. Implemented in Verilog, this module is designed for efficient execution on FPGA hardware, ensuring that computations adhere to constraints related to numerical precision, latency, and resource utilization. The fully connected architecture of the ANN requires each neuron to process multiple inputs, making an optimized neuron module crucial for achieving real-time handwritten character recognition

A. Architecture of the Neuron

The neuron is parameterized to allow configurability for layer and neuron numbers, data width, activation type (ReLU or Sigmoid), and file-based initialization of weights and biases. Its high-level functionality can be summarized as:

• Accepting input values (Xi), weights (Wi), and biases (B) through a memory interface.

• Performing the weighted sum:

$$Z = \sum_{i=1}^{N} X_i . W_i + B$$

• Applying an activation function:

A = ReLU(Z) = max(0,Z)

OR

A= $\frac{1}{1+e^{-z}}$ $(Sigmoid)$

• Outputting the activation result (A) for the next layer.

For an output layer with nnn neurons, where each neuron produces a logit z_i, the softmax function is:

$$\sigma(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{n} e^{z_j}}$$

Where:

- z_i is the raw output (logit) from neuron i.

- e^{z_i} applies the exponential function to amplify differences.
- The denominator ensures that all outputs sum to **1**, converting them into probabilities.

(A) Compute Logits from Fully Connected Layer

- The final fully connected layer produces raw scores (logits) based on weighted sums:

$z_i = W_i \cdot \mathrm{X} + b_i$

Where:

- w_i = Weights for output neuron iii.
- X= Input feature vector from the previous layer.
- b_i = Bias for output neuron iii.

(B) Exponentiate Each Logit

- Apply exponentiation e^{z_i} to amplify larger values.

(C) Compute the Sum of All Exponentiated Values

- Compute:

$$S = \sum_{j=1}^{n} e^{z_j}$$

This acts as a normalizing factor.

(D) Normalize Each Output

- Compute the softmax probability:

$$\sigma(z_i) = \frac{e^{z_i}}{S}$$

- This ensures that all outputs sum to 1.

Softmax function:

Probability Distribution: Converts logits into a probability-like interpretation.
Multi-Class Classification: Ensures only one class dominates with the highest probability.
Differentiable: Works well with backpropagation for training.

B. Hardware Implementation

The Verilog implementation uses memory to store weights and biases, and a pipeline structure ensures efficient computation:

• Memory Initialization: Weights and biases are loaded at runtime or initialized using predefined files for pretrained models. The weight memory block is parameterized for neuron count and layer depth.

• Weighted Sum Computation: Input values are multiplied with weights using a signed multiplier. Intermediate results are stored in a 32-bit register to prevent overflow during summation.

• Activation Function: A module implements ReLU using conditional logic. For Sigmoid, a ROM-based lookup table is used to approximate the non-linear function efficiently.

C. Neuron Realization on Zynq Zed Board FPGA

Figure 3 illustrates the realized neuron module implemented on the Zynq Zed Board FPGA. The design leverages DSP48E1 blocks for multiplications and additions, BRAM18E1 for storing weights and biases, and a pipeline structure for efficient computation. The module processes inputs with fixed-point arithmetic, ensuring numerical stability and hardware efficiency. The DSP blocks perform the arithmetic operations, while BRAM is used for weight and bias storage. The pipeline architecture ensures high throughput by overlapping computations across clock cycles. The neuron interacts with other layers by propagating activation results to subsequent layers via a pre-defined control flow. This modular approach facilitates scalability, allowing the design to be extended for larger networks.

D. Numerical Precision and Overflow Handling

To ensure numerical stability:

• Fixed-point arithmetic is used for inputs, weights, and biases. For example, a Q15 format (16-bitrepresentation) balances precision and hardware efficiency.

• Overflow is mitigated by extending intermediate registers to 32 bits and saturating values that exceed the representable range.

• Quantization errors are minimized by scaling weights during training, ensuring compatibility with the fixedpoint representation in hardware.

E. Resource Optimization and Trade-offs Increasing the data width from 16 bits to 32 bits reduces quantization errors but increases FPGA resource utilization (e.g., LUTs and DSPs) and power consumption. Table I summarizes the trade-offs.

Table I: Impact of Data Width on Resource Utilization

Data Width	LUTs	DSPs
16-bit	12,345	32
32-bit	24,600	64

F. Integration and Testing

The neuron module was integrated with layer modules to form a fully connected network. A Verilog testbench was used to validate its functionality, with inputs derived from the MNIST dataset. The outputs were compared against a software implementation for accuracy validation.

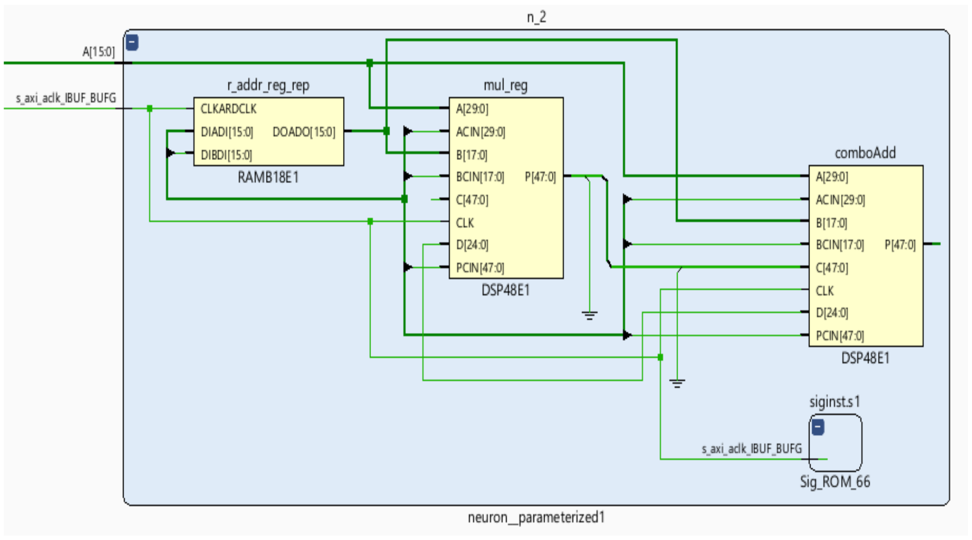

Fig. 3. Neuron Realization on Zynq Zed Board FPGA.

VI. ILLUSTRATIVE EXAMPLE OF DATA FLOW AND COMPUTATION

This section demonstrates the data flow and computation through a numerical example of weight initialization, forward propagation, and Verilog implementation.

A. Numerical Example of Neuron Computation

Consider a single neuron in the first layer:

• Input Vector: X = [0.5, 0.2, 0.8]

• Weight Vector: W= [0.3, 0.7, 0.1]

• Bias: B = 0.5

• Activation Function: ReLU(x) = max(0, x)

The computation involves two steps:

1) Weighted Sum Calculation:

$$Z = \sum_{i=1}^{3} X_i \,.\, W_i + B$$

$$= (0.5 \times 0.3) + (0.2 \times 0.7) + (0.8 \times 0.1) + 0.5$$

$$= 0.88$$

2) Activation Output:

$$A = \text{ReLU}(Z) = \max(0, 0.88) = 0.88$$

B. Overflow Handling and Edge Cases

To ensure numerical stability and avoid overflow:
• Intermediate weighted sums (Z) are stored in a 32-bit signed register to handle large values during multiplication.
• Fixed-point representation (e.g., Q15 format) is used for inputs and weights to balance precision and hardware efficiency.
• Activation outputs are truncated to 16 bits to match hardware constraints and downstream compatibility.
• During training, weights and biases are scaled to minimize the risk of overflow in hardware operations.

Impact on Accuracy: Overflow handling ensures that the model remains stable during both training and inference.Quantization errors are mitigated through proper scaling, preserving the model's accuracy.

C. Visualization of Data Flow

The block diagram in Figure 4 visually represents the data flow between neurons, layers, and their connections, emphasizing the modularity and scalability of the design. The block diagram represents the data flow of an FPGA-based artificial neural network (ANN) for handwritten character recognition using a fully connected architecture. The input data, likely pixel values from a preprocessed character image, is fed into Layer 1, where initial computations occur. Subsequent layers (Layer 2, Layer 3, and Layer 4) process the data through weighted connections, with intermediate registers storing activation values. Instead of using a computationally expensive Softmax function, the maxFinder block determines the most probable class using an Argmax operation, which efficiently selects the neuron with the highest activation. The final classification result is passed through the AXI Lite interface, making it accessible to an external processor or memory. The design leverages parallel computation and modularity, ensuring scalability and efficiency for FPGA implementation in Vivado.

VII. RESULTS AND ANALYSIS

A. Resource Utilization for ReLU and Sigmoid Activation Functions

Tables II summarize the FPGA resource utilization for ReLU and Sigmoid activation functions, respectively. This data highlights the trade-offs between operational frequency, resource utilization, and overall accuracy of the two activation functions.

Table II: Resource Utilization For ReLU And Sigmoid Activation Functions

Resources	ReLU Utilization	Sigmoid Utilization	ReLU Utilization(%)	Sigmoid Utilization(%)
LUT	130	71	0.24	0.13
FLIP-FLOPS	66	52	0.06	0.05
BRAM	0.5	1	0.36	0.71

DSP	2	2	0.91	0.71
I/O	36	36	18.0	18.0
BUFG	1	1	3.13	3.13
Fmax(MHz)	210	194	-	-

B. Performance Comparison Between ReLU and Sigmoid

The performance of the ReLU and Sigmoid activation functions was evaluated based on FPGA resource utilization, operational frequency, and recognition accuracy. The results are summarized below:

- ReLU: - Higher operational frequency of 210 MHz. -Lower utilization of FPGA resources. - Suitable for real-time, high-speed applications. - Accuracy: 96%.

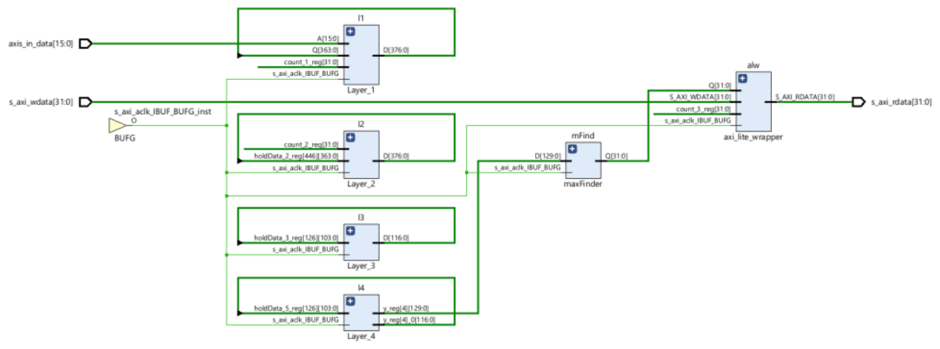

Fig. 4. Block Diagram Showing Data Flow Between Layers and Neurons.

Table III: Performance Comparison Between ReLU And Sigmoid Activation Functions

Performance Metric	**ReLU**	**Sigmoid**
Operational Frequency(MHz)	210	194
FPGA Resource Utilization	Lower	Slightly Higher
Application Suitability	Real-time,High-Speed Applications	Precision-Critical Tasks
Accuracy	96%	98%

- Sigmoid: - Operational frequency of 194 MHz. – Slightly higher resource utilization compared to ReLU. - Achieves a superior accuracy of 98%, making it ideal for precision-critical tasks.

Figure 5 visualizes the trade-offs between resource utilization and operational frequency for the two activation functions.

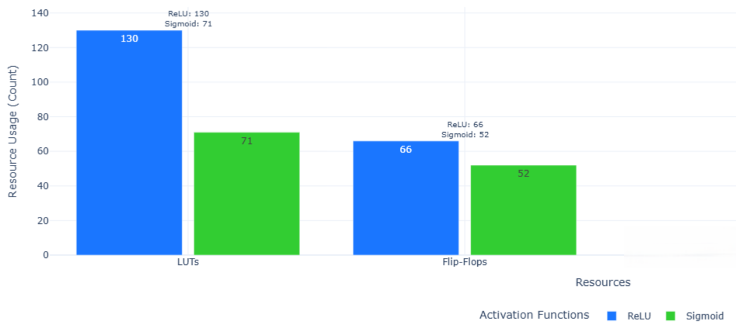

Fig. 5. fix the size of the chart Comparison of Resource Utilization and Performance Metrics for ReLU and Sigmoid Activation Functions.

C. Analysis

This comparison shows that ReLU is simpler and faster to compute than Sigmoid, utilizing fewer logic and memory resources; however, Sigmoid can have a recognition accuracy of 98%, critical in applications demanding maximal precision.

The resource utilization metrics, especially the DSP efficiency, are in line with works like QFX [7], which, by using multiplier-free quantization techniques, seeks to lower the usage of DSPs. Such techniques further lower FPGA resource consumption while keeping accuracy and performance measures competitive.

VII. CONCLUSION

This work presents a Verilog-based Artificial Neural Network (ANN) for FPGA handwritten letter recognition. It features a 5-layer fully connected architecture using ReLU and Sigmoid for efficient pattern recognition. The design optimizes resource consumption, ensuring real-time performance with minimal DSP and memory usage. FPGA parallelism enhances speed while fixed-point arithmetic and BNNs reduce computational complexity. Comparisons with floating-point ANNs highlight improvements in efficiency and hardware feasibility. Lessons from Perceptron models and VHDL-based ANNs guide weight optimization and execution improvements. Trained on the MNIST dataset, the model achieves 98% accuracy, validated through Vivado testing. The approach demonstrates FPGA-based ANNs' potential for low-power, real-time character recognition. Future enhancements include hybrid ANN-Perceptron models and improved weight accuracy optimization. This research contributes to hardware-accelerated AI, offering a flexible and efficient FPGA-based solution.

REFERENCES

[1] Sumayyabeevi, V. A., Jaimy James Poovely, N. Aswathy, and S. Chinnu. "A new hardware architecture for fpga implementation of feed forward neural networks." In 2021 2nd International Conference on Advances in Computing, Communication, Embedded and Secure Systems (ACCESS), pp. 107-111. IEEE, 2021.

[2] A. S. Kumar, S. R. Pillutla, T. V. Babu, A. N. Kabra, P. Kavyasree and M. Rahul, "A Novel Approach to Implement a 4-Bit Vedic Multiplier Using Reversible Techniques," 2024 15th International Conference on Computing Communication and Networking Technologies (ICCCNT), India, pp. 1-6, 2024.

[3] Aruru Sai Kumar et al., "Principle Study of MoS2 FET at lower Channel Lengths." In Journal of Physics: Conference Series, vol. 2837, p. 012080. 2024.

[4] C Ganesh et al., "A Novel Design of Area Efficient Full Adder Architecture using Reversible Logic Gates",7th International Conference on Devices, Circuits, and Systems (ICDCS), 2024.

[5] A. S. Kumar, R. Kusuma, A. Ramya, D. Mounika, V. Gouda and K. B. K. Sri Nithin, "A Novel Reversible Gate Design for Comprehensive Logic and Arithmetic Operations," 2024 5th International Conference on Electronics and Sustainable Communication Systems (ICESC), Coimbatore, India, pp. 75-80, 2024.

[6] A. S. Kumar, R. P. Somineni, B. N. Kumar Reddy and A. K. Varasala, "Enhancing Lightweight Embedded Systems with Network-on- Chip Based Memory Organization Unit," 2024 5th International Conference on Electronics and Sustainable Communication Systems (ICESC), India, pp. 126-130, 2024.

[7] A. S. Kumar, N. Lakshman Pratap, N. Neelima, T. C. Mahajani, V. Kalyan Kumar and P. Anunya, "An Optimised Design of a 4-bit Carry Look-Ahead Adder Using Novel Reversible Logic Gates," 2024 International Conference on Intelligent Algorithms for Computational Intelligence Systems (IACIS), Hassan, India, pp. 1-6, 2024.

[8] A. S. Kumar, A. Ramya, U. Siddhesh, N. Y. Meera, T. Kandimalla and P. Abhishek, "A Comparator Tree Design with Optimized Quantum Cost Using Reversible Logic Techniques," 2024 International Conference on Intelligent Algorithms for Computational Intelligence Systems (IACIS), Hassan, India, pp. 1-6, 2024.

[9] Aruru Sai Kumar et al., "Gate Dielectric Engineering on 2D FETs for Continued Scaling." In Journal of Physics: Conference Series, vol. 2837, no. 1, p. 012051. IOP Publishing, 2024.

[10] A. S. Kumar, N. L. Pratap, A. Ramya, V. Upendra, E. Venugopal and P. Abhishek, "An Efficient BCD Adder Design Utilizing Reversible Logic Techniques," 2024 International Conference on Intelligent Algorithms for Computational Intelligence Systems (IACIS), Hassan, India, pp. 1- 5, 2024.

[11] A. S. Kumar, P. Santhosh, N. Neelima, B. V. Harshitha, D. S. A. Lakshmi and K. S. Vardhan, "A Novel ASK Reversible Gate and its Implementation in Ripple Carry Adder Design," 2024 IEEE International Conference of Electron Devices Society Kolkata Chapter (EDKCON), India, pp. 1-6, 2024.

[12] Aruru Sai Kumar, T.V.K. Hanumantha Rao, “Scalable benchmark synthesis for performance evaluation of NoC core mapping”, Microprocessors and Microsystems, Vol. 79, 2020.

[13] Aruru Sai Kumar, T.V.K. Hanumantha Rao, “An Adaptive Core Mapping Algorithm on NoC for Future Heterogeneous System-on-Chip”, Computers and Electrical Engineering, Vol. 95, 2021.

[14] A. S. Kumar et al., ”Device Analysis of Vertically Stacked GAA Nanosheet FET at Advanced Technology Node,” 2023 3rd International Conference on Advances in Computing, Communication, Embedded and Secure Systems (ACCESS), India, pp. 274-279, 2023.

[15] Kumar, Aruru Sai, et al., “Nanosheet FET for Future Technology Scaling”, Integrated Devices for Artificial Intelligence and VLSI, pp. 25-47, 2024.

[16] Aruru Sai Kumar et al., “Nanosheet Field Effect Transistor Device and Circuit Aspects for Future Technology Nodes,”ECS Journal of Solid State Science and Technology, Vol. 12, No. 8, 2023.

[17] Aruru Sai Kumar, U. Siddhesh, N. Sai kiran and K. Bhavitha, “Design of High Speed 8-bit Vedic Multiplier using Brent Kung Adders,” 2022 13th International Conference on Computing Communication and Networking Technologies (ICCCNT), Kharagpur, India, pp. 1-5, 2022.

[18] C.Sarvan and M. Gunduzalp, “Implementation of ANN Training Module on Field Programmable Gate Arrays,” 2019 Innovations in Intelligent Systems and Applications Conference (ASYU), Izmir, Turkey, 2019, pp. 1-6, doi: 10.1109/ASYU48272.2019.8946350.

[19] S. Saini, R. S. Komaragiri, S. S. Chouhan and M. Kumawat, “Implementation of XOR Gate using AOI model by Reconfigurable Artificial Neural Network on FPGA,” 2024 15th International Conference on Computing Communication and Networking Technologies (ICCCNT), Kamand, India, 2024, pp. 1-7, doi: 10.1109/ICCCNT61001.2024.10726252

[20] Hayman, Samantha. “The McCulloch-Pitts model.” IJCNN’99. International Joint Conference on Neural Networks. Proceedings (Cat. No.99CH36339) 6 (1999): 4438-4439 vol.6.

[21] G. Strawn, ”Where Deep Learning and Generative AI Started: Masterminds of Artificial Neural Networks—McCulloch, Pitts, and Rosenblatt” in IT Professional, vol. 26, no. 03, pp. 99-101, May-June 2024, doi: 10.1109/MITP.2024.3404229.

[22] A. Awasthi, Pranjal, A. Chaturvedi, V. Shukla and M. K. Misra, ”Mathematics and Logics in ML: Application Aspects,” 2024 IEEE International Students’ Conference on Electrical, Electronics and Computer Science (SCEECS), Bhopal, India, 2024, pp. 1-6, doi: 10.1109/SCEECS61402.2024.10482143.

[23] R. Sinha, S. K. Gupta, P. Rajalingam and S. Routray, ”Detecting Handwritten digits using fully connected Artificial Neural Network in FPGA which will act like a replacement for GPU,” 2023 International Conference on Recent Advances in Electrical, Electronics, Ubiquitous Communication, and Computational Intelligence (RAEEUCCI), Chennai, India, 2023, pp. 1-5, doi: 10.1109/RAEEUCCI57140.2023.10134313.

[24] D. Dai, Y. Zhang, J. Zhang, Z. Hu, Y. Cai, Q. Sun, and Z. Zhang, ”Trainable Fixed-Point Quantization for Deep Learning Acceleration on FPGAs,” 2021 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), pp. 1-8, doi: 10.1109/ICCAD51445.2021.9731511.

[25] Wang, Peng Song, Jiayu Peng, Yinquan Liu, Guipeng. (2020). Binarized Neural Network Based On FPGA To Realize Handwritten Digit Recognition. 1204-1207. 10.1109/ICIBA50161.2020.9276909.