Grenze International Journal of Engineering and Technology, January Issue

Pattern Recognition on Neural Networks using FPGA

Dr. Abhay Chopde[1], Dr. Vijay Mane[2], Sarthak Pandit[3], Hitesh Pariani[4] and Kartik Parsodkar[5]
[1-5]Department of Electronics and Telecommunication Engineering, Vishwakarma Institute of Technology, Pune, India
Email: {sarthak.pandit21, abhay.chopde, vijay.mane, hitesh.pariani21, kartik.parsodkar21}@vit.edu

***Abstract*—Pattern recognition is crucial in various applications, including biometrics, speech recognition, and image processing. With advancements in technology, the demand for efficient and high-speed systems has increased. Field-Programmable Gate Arrays (FPGAs) are a promising platform for implementing pattern recognition algorithms due to their reconfigurability, parallelism, and low power consumption. This research paper explores the design, optimization, and performance evaluation of neural network-based pattern recognition systems on FPGA platforms. Key focus areas include selecting and optimizing neural network models, leveraging parallelism and pipelining techniques, and addressing resource constraints and power efficiency. The study uses a systematic approach, including theoretical analysis, simulation studies, and practical implementation, to validate proposed methodologies and algorithms. The findings are expected to contribute significantly to digital design in electronics and telecommunication engineering.**

***Index Terms*— Field Programmable Gate Arrays, Neural Networks, Pattern Recognition, Machine Learning, Graphical Processing Units.**

I. INTRODUCTION

Pattern recognition, a fundamental aspect of artificial intelligence and machine learning, pertains to the automated identification of patterns or regularities in data. It encompasses a diverse array of applications, from biometric authentication and speech recognition to medical diagnosis and image processing. In engineering disciplines, pattern recognition serves as a cornerstone for numerous tasks, including fault detection in machinery, signal classification in telecommunications, and object recognition in computer vision. The imperative for pattern recognition stems from the burgeoning volumes of data generated across various domains, necessitating efficient and reliable methods for extracting meaningful insights. Traditional computing architectures often struggle to cope with the computational demands inherent in pattern recognition tasks, particularly those involving complex datasets or real-time processing requirements. Field-Programmable Gate Arrays (FPGAs) emerge as a compelling solution to address the computational challenges associated with pattern recognition. FPGAs offer unique advantages, including reconfigurability, parallelism, and hardware-level customization, making them well-suited for accelerating pattern recognition algorithms. Unlike traditional processors, FPGAs allow for the implementation of custom hardware architectures tailored to specific pattern recognition tasks, thereby enabling significant gains in performance and efficiency.

By harnessing the parallel processing capabilities of FPGAs, pattern recognition algorithms can be executed in parallel across multiple processing elements, resulting in dramatic speed enhancements compared to sequential execution on conventional processors. Moreover, the inherent parallelism of FPGAs enables the simultaneous processing of multiple data streams, facilitating real-time pattern recognition in applications where timely

Grenze ID: 01.GIJET.11.1.820

decision-making is paramount.

II. Literature Survey

Convolutional Neural Networks (CNNs) have become ubiquitous in various machine learning applications due to their ability to achieve high inference performance. Fully binarized CNNs, utilizing single-bit weights and activations with XNOR type operators, have emerged as a promising approach for efficient inference on hardware platforms. However, [1] indicates that full binarization often leads to a degradation of accuracy, prompting the exploration of alternative strategies to mitigate this issue. [1], [2] and [3] involves the development of a multi-precision CNN framework that integrates both binarized and floating-point CNNs in a pipeline configuration deployed on heterogeneous hardware. This approach aims to combine the efficiency of binarized CNNs with the accuracy of floating-point networks, achieving a flexible trade-off between accuracy and throughput. The proposed multi-precision CNN inference approach operates by running the full dataset and a subset of it in reduced and full precision implementations, respectively, in parallel. A Dynamic Model Update (DMU) function is employed to adjust the trade-off between accuracy and speed by determining the degree of subset recalculation. This mechanism enables the system to adapt dynamically based on the classification confidence level, thereby optimizing both accuracy and throughput.

Experimental results presented in the research paper demonstrate the effectiveness of the multi-precision concept in improving both binarized CNN accuracy and floating-point network throughput. Specifically, the multi-precision network deployed on a Zynq 7020 device showcases significant improvements, increasing the binarized CNN accuracy from 78.5% to 82.5% and the CPU inference speed from 29.68 to 90.82 images/sec. Despite these promising results, the tested configuration faces limitations, primarily due to the low throughput achieved in the Cortex A9 processors. Future work aims to extend the multi-precision concept to higher-end heterogeneous devices, incorporating ARMv8 processors with active NEON engines. Additionally, the exploration of mixed precision techniques on FPGA hardware is considered to further enhance the efficiency and performance of the proposed approach.

Super-resolution systems play a vital role in enhancing the quality of images or video by generating high-resolution renditions from low-resolution inputs using computational algorithms. The development of Convolutional Neural Networks (CNNs) and Deep Learning (DL) methods in [4] has revolutionized super-resolution tasks, surpassing traditional methods in performance. However, the increasing depth and complexity of CNN models have led to a surge in the number of parameters, rendering them impractical for resource-constrained environments like smartphones or mobile systems. In response to these challenges, the research has focused on optimizing super-resolution models to reduce computational costs and memory requirements. The paper titled "ResBinESPCN" proposes an efficient binary neural network architecture specifically tailored for image super-resolution tasks. The ResBinESPCN architecture improves energy efficiency through algorithmic and hardware-level optimizations, enhancing computational efficiency and reducing memory consumption. Experimental validation demonstrates the effectiveness of the proposed architecture, including ablation studies on models with varying data bit widths.

One notable aspect of the ResBinESPCN model is its use of Binary Neural Networks (BNNs) to accelerate computations and reduce memory consumption. By binarizing the data and employing low bit-width data types, the model achieves significant improvements in computational efficiency without sacrificing performance. Additionally, the introduction of a shortcut structure enhances reconstruction quality, further boosting the model's effectiveness in image super-resolution tasks. Furthermore, the paper discusses the deployment of the ResBinESPCN model on Field-Programmable Gate Arrays (FPGAs) using the FINN framework

Binary neural networks (BNNs) represent a subset of artificial/deep neural network (ANN/DNN) architectures that leverage binary values for weight representation, thereby accelerating both training and inference processes while reducing hardware complexity and model sizes. BNNs are particularly suitable for implementation in resource-constrained devices like Field-Programmable Gate Arrays (FPGAs) and Application-Specific Integrated Circuits (ASICs). However, the binarization process often leads to a tradeoff between performance and accuracy, prompting ongoing research efforts to bridge this gap. [5] provides a comprehensive review of BNNs for FPGA implementation, covering various aspects including BNN architectures and variants, design and tool flows for FPGAs, and diverse applications for BNNs. Despite their advantages in model compression and inference speed, BNNs typically exhibit lower accuracy compared to full-precision DNNs. Nevertheless, recent advancements in BNN research have focused on narrowing this accuracy gap through various techniques. One approach involves the incorporation of gain terms or learnable gain terms into BNN architectures, enabling better representation of weights and improving accuracy without sacrificing speed or model size. Despite these

challenges, the emergence of tool flows such as FINN and HLS4ML has simplified the process of implementing BNNs and Quantized Neural Networks (QNNs) on FPGA platforms, making them more accessible for designers. [6] outlines the intrinsic parallel structure of neural networks, which not only enables swift computation of specific tasks but also renders them highly suitable for implementation in VLSI (Very Large Scale Integration) technology. However, the hardware realization of neural networks, particularly those with a large number of neurons, remains a formidable challenge. FPGA-based reconfigurable computing architectures emerge as a promising solution for this challenge due to their inherent flexibility and programmability. The paper presents a hardware design of an artificial neural network on FPGA platforms, specifically focusing on realizing a feedforward multilayer neural network. Leveraging VHDL (Very High Speed Integrated Circuits Hardware Description Language), the designed architecture aims to efficiently implement neural network functionality on FPGA devices.

[7] introduces a pioneering FPGA System-on-Chip (SoC) implementation for the challenging task of recognizing handwritten multi-digit numbers. Unlike traditional approaches reliant on computationally-intensive segmentation methods, the proposed solution adopts a novel digit extraction technique based on identifying non-zero columns in images. This innovative approach offers a streamlined alternative to segmentation, potentially reducing computational overhead and enhancing efficiency. In addition to digit extraction, the paper employs a multi-layer neural network for digit prediction, leveraging the capabilities of FPGA hardware for accelerated computation. The authors provide a detailed account of the design and FPGA implementation process, highlighting various optimization techniques employed to enhance performance. These optimizations contribute to a significant speed-up compared to software-based solutions, while achieving an impressive detection accuracy of 96.76%.

The implementation of multilayer feedforward neural networks (NNs) using field-programmable gate arrays (FPGAs) has garnered significant attention due to its potential in various applications. [8] addresses the challenge of resource utilization in FPGA-based NN implementations, focusing on reducing resource requirements while maintaining computational efficiency. Previous research has explored FPGA-based NN implementations to leverage the parallel processing capabilities of FPGAs for accelerating neural network computations. However, the limitation lies in the finite resources available on FPGA chips, particularly the abundance of multipliers required for NN operations. This limitation restricts the size of NNs that can be implemented on a single FPGA, thereby hindering their commercial viability. To overcome this challenge, the authors propose a novel approach called layer multiplexing, which exploits the sequential processing nature of NN layers. Instead of implementing the entire network, only the single largest layer is instantiated on the FPGA, and a control block manages the computation of different layers. This method aims to reduce resource requirements while maintaining computational speed, thus enabling the realization of larger NNs on a single FPGA at a lower cost. The concept of layer multiplexing is supported by the implementation of multilayer feedforward NNs on Xilinx FPGA "XCV400hq240."

While layer multiplexing offers significant resource savings, it introduces a moderate overhead on computational speed due to the increased complexity of the control block, particularly as the number of layers in the network grows. However, given the high operating clock frequency of FPGA devices, the impact on computational time per sample is deemed acceptable for many applications. To validate the proposed technique, the authors develop and test an NN-based flux estimator for sensorless drives. The results showcase the efficacy of the approach, demonstrating its feasibility and promising potential in real-world applications.

III. METHODOLOGY

The methodological framework employed in this investigation represents a systematic and comprehensive approach to deploying a neural network on a Field-Programmable Gate Array (FPGA). A key preprocessing step entails normalization, wherein the pixel values of each image are rescaled to fall within the range of [0, 1]. This normalization process facilitates uniformity in data representation and aids in mitigating issues related to varying illumination conditions or contrast levels present in the original images. Normalization plays a pivotal role in enhancing the robustness and convergence properties of neural network models during training. By constraining pixel values within a standardized range, the neural network can effectively learn and generalize patterns without being unduly influenced by variations in pixel intensity. Furthermore, normalization aligns with the underlying assumptions of many machine learning algorithms, which often operate optimally when input features are scaled or normalized to comparable magnitudes.

The architecture is structured as a feedforward neural network, a widely used paradigm in the field of deep learning. This architecture is chosen for its simplicity, effectiveness, and scalability, making it suitable for a broad range of applications.

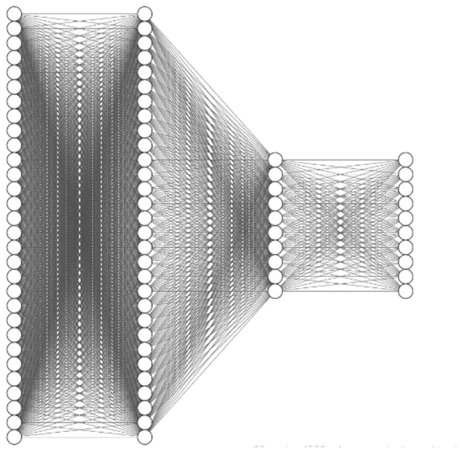

Fig. 1 Feed Forward Neural Network Structure

The neural network consists of multiple layers, each comprising interconnected neurons responsible for processing and transforming input data. In this particular architecture, the input layer is designed to accommodate the features extracted from the preprocessed data. With 30 neurons in the input layer, the network can effectively capture and represent the relevant characteristics of the input patterns.

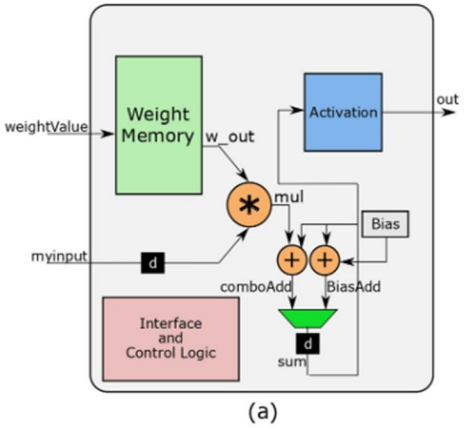

Fig. 2 Internal implementation of single neuron

Subsequently, two hidden layers are incorporated into the architecture, each housing 30 neurons. These hidden layers serve as intermediary stages in the neural network, enabling the extraction of increasingly abstract features from the input data. The inclusion of multiple hidden layers allows for greater flexibility in capturing complex patterns and relationships present in the dataset. Finally, an output layer consisting of 10 neurons is integrated into the architecture to produce the final predictions. The number of neurons in the output layer corresponds to the number of classes or categories in the pattern recognition task. Each neuron in the output layer is associated with a specific class, and the network's output is interpreted as the probability distribution over these classes. By applying Sigmoid activation functions, the neural network can effectively model nonlinear mappings between input and output spaces, thus enhancing its capacity to perform accurate pattern recognition tasks. Leveraging the computational prowess of Graphics Processing Units (GPUs), the designed neural network undergoes Training. Utilizing backpropagation algorithms and optimization techniques such as stochastic gradient descent or advanced methods like Adam, the network iteratively refines its parameters until convergence or satisfactory performance on validation datasets is achieved. Following the training phase, the subsequent step in the methodology involves Weight Quantization and Conversion. In this critical step, the trained weights, typically represented in floating-point format, are converted into a more compact 16-bit fixed-point representation. This

quantization process is imperative for efficient deployment of the neural network on resource-constrained hardware platforms such as FPGAs. The quantization strategy employed preserves one bit for sign representation, allocating five bits for the integer component, and reserving ten bits for the decimal component. This allocation scheme ensures that the quantized weights maintain a sufficient level of precision while mitigating potential precision loss associated with the conversion process. By reducing the bit-width of the weights, the storage requirements and computational complexity of the neural network are significantly reduced, facilitating efficient implementation on hardware platforms.

Graph | **Table**

Resource	Utilization	Available	Utilization %
LUT	70	41000	0.17
FF	52	82000	0.06
BRAM	1	135	0.74
DSP	2	240	0.83
IO	36	300	12.00
BUFG	1	32	3.13

Fig.3 Resource utilization of Sigmoid Function

To efficiently approximate the Sigmoid activation function, a Lookup Table Implementation approach is adopted. This strategy involves the meticulous design of a lookup table covering the entire range of possible inputs with adequate resolution. Implementation strategies focus on hardware efficiency, aiming to minimize resource utilization through techniques such as Look-Up Tables (LUTs) and Flip-Flops (FF). Subsequently, the designed architecture, along with the converted weights and the Sigmoid lookup table, is deployed on FPGA platforms using FPGA development tools such as Vivado or Quartus. Extensive optimization efforts are channelled towards enhancing both performance and resource utilization metrics. The evaluation phase rigorously assesses the FPGA-based neural network's accuracy and inference speed. Comparative analysis against the original GPU-trained model is conducted to discern performance discrepancies. Furthermore, experiments probing the impact of weight quantization and Sigmoid approximation on accuracy and resource utilization are performed, yielding comprehensive insights into the efficacy of the FPGA-based implementation.

IV. Results and Discussions

After the extensive experimentation phase, the culmination of our research journey brings forth the presentation and critical analysis of the results obtained from deploying the neural network on both FPGA and GPU platforms. This pivotal phase entails a thorough examination of various performance metrics, with accuracy serving as the primary focus, thereby offering profound insights into the comparative effectiveness of the FPGA-based implementation as opposed to its GPU counterpart. The FPGA-based neural network, upon testing, exhibited an accuracy rate of 91%. In contrast, the model trained on GPU displayed a slightly superior accuracy of 95%. The obtained results underscore the successful deployment of the neural network architecture on the FPGA platform. Despite a marginal reduction in accuracy compared to the GPU-trained model, the FPGA-based implementation showcases a commendable level of proficiency in pattern recognition tasks. This reduction in accuracy, albeit slight, is a noteworthy observation deserving of meticulous examination. While GPU platforms may exhibit slightly superior accuracy in certain scenarios, the FPGA-based approach offers distinct advantages in terms of scalability, power efficiency, and resource utilization.

Furthermore, the optimization strategies employed for each platform significantly influence their respective performance levels. GPU implementations often benefit from highly optimized libraries and frameworks tailored specifically for deep learning tasks, such as TensorFlow and PyTorch. These frameworks leverage advanced optimization techniques and parallel processing capabilities inherent in GPUs, thereby maximizing computational efficiency and accuracy. In contrast, FPGA-based implementations necessitate meticulous hardware-level optimization and algorithmic design to harness the platform's parallelism effectively. The effectiveness of these optimization strategies profoundly influences the accuracy and efficiency of the deployed neural network models on FPGA platforms. Additionally, the memory hierarchy and bandwidth limitations inherent in FPGAs pose unique challenges in efficiently managing data movement and computation

A. Resource Constraints on FPGA

FPGA platforms inherently impose resource constraints compared to their GPU counterparts, necessitating meticulous optimization efforts during implementation. Despite these efforts, compromises may have been required, potentially impacting the fidelity of the deployed neural network. Balancing resource utilization with computational demands is critical to achieving optimal performance within the constraints imposed by FPGA architectures, underscoring the importance of strategic resource management in FPGA-based implementations of neural networks.In the proposed system, the DeepLabV3+ model achieved an IoU score of 0.79. This indicates that, on average, 79% of the predicted flood areas correctly overlapped with the actual flood extents. While this is a commendable result, it also suggests there is room for improvement. An IoU score of 0.79 is generally considered strong for semantic segmentation tasks, particularly given the complexities inherent in aerial imagery, such as varying scales, resolutions, and background clutter.

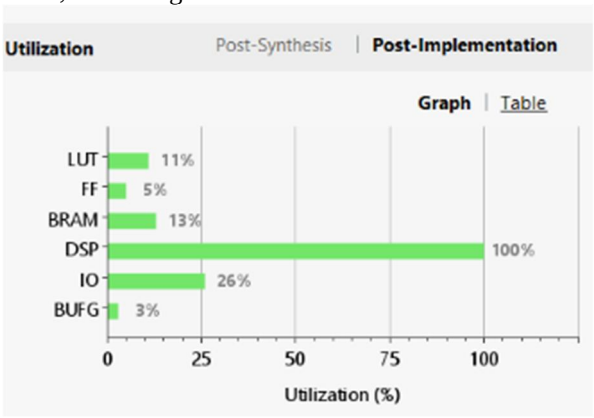

Fig. 4 Resource Utilization obtained in proposed system

B. Precision Loss due to Quantization

The process of weight quantization introduces precision loss as floating-point values are converted to fixed-point representation. Despite careful attention to minimizing this loss, subtle inaccuracies may propagate during quantization, affecting the accuracy of the deployed neural network. Quantization strategies must strike a delicate balance between reducing computational complexity and preserving model fidelity, emphasizing the need for rigorous validation and optimization techniques to mitigate the impact of precision loss.

C. Approximation of Sigmoid Function

The utilization of a Sigmoid lookup table for activation function approximation on FPGA aims to minimize resource utilization while maintaining computational efficiency. However, the approximation introduced by the lookup table may deviate marginally from the ideal Sigmoid function, potentially influencing the accuracy of the neural network, albeit to a limited extent. Optimizing the lookup table design and calibration parameters is crucial to minimizing approximation errors and ensuring consistency in model performance.

D. Inference Speed and Latency

Beyond accuracy, inference speed and latency are paramount considerations, particularly in real-time applications. Despite potentially lower accuracy, FPGA-based implementations may offer superior inference speed and reduced latency due to parallelism and hardware-level optimization. Leveraging FPGA architectures' inherent advantages in real-time processing and low-level hardware customization enables efficient execution of neural network computations, highlighting the trade-offs between accuracy and computational efficiency in FPGA-based deployments.

E. Scalability and Deployment Considerations

While FPGA-based implementations may exhibit slightly lower accuracy compared to GPU counterparts, their scalability and deployment flexibility offer unique advantages. With power efficiency, real-time processing capabilities, and reconfigurability, FPGAs are well-suited for resource-constrained and latency-sensitive environments. Strategic deployment of neural networks on FPGA platforms requires careful consideration of application requirements and optimization techniques to leverage FPGA architectures' strengths effectively, emphasizing the importance of tailored solutions for diverse deployment scenarios.

V. Conclusion

In conclusion, the deployment of a neural network on a Field-Programmable Gate Array (FPGA) platform presents a promising avenue for accelerating inference tasks, particularly in resource-constrained and latency-sensitive environments. Through a meticulously orchestrated methodology encompassing data preprocessing, neural network architecture design, training on GPU, weight quantization and conversion, Sigmoid lookup table implementation, FPGA deployment, and subsequent evaluation, insights have been gleaned into the efficacy of FPGA-based implementations compared to their GPU counterparts. The results of the experimentation reveal that while the FPGA-based neural network achieved a commendable accuracy of 91% during testing, a marginal discrepancy exists compared to the GPU-trained model, which exhibited a slightly higher accuracy of 95%. This variance in accuracy can be attributed to several factors, including resource constraints on FPGA, precision loss due to weight quantization, approximation of the Sigmoid function, and considerations pertaining to inference speed and latency.

Despite the observed variance in accuracy, the FPGA-based implementation offers compelling advantages, including enhanced resource efficiency, real-time processing capabilities, and scalability. These attributes render FPGA platforms well-suited for deployment in diverse applications, particularly those necessitating low-latency inference and power-efficient computing. In light of these findings, future research endeavors may focus on further optimizing FPGA-based implementations to bridge the accuracy gap while leveraging the unique advantages offered by FPGA architectures. Additionally, exploring advanced optimization techniques, refining approximation strategies for activation functions, and investigating novel architectures tailored to FPGA platforms hold promise for enhancing the performance and applicability of neural network deployments on FPGA.

References

[1] Amiri, S., Hosseinabady, M., McIntosh-Smith, S., & Nunez-Yanez, J. (2018, March). Multi-precision convolutional neural networks on heterogeneous hardware. In 2018 Design, Automation & Test in Europe Conference & Exhibition (DATE) (pp. 419-424). IEEE.

[2] Su, Y., Seng, K. P., Smith, J., & Ang, L. M. (2024). Efficient FPGA Binary Neural Network Architecture for Image Super-Resolution. Electronics, 13(2), 266.

[3] Su, Y., Seng, K. P., Ang, L. M., & Smith, J. (2023). Binary Neural Networks in FPGAs: Architectures, Tool Flows and Hardware Comparisons. Sensors, 23(22), 9254.

[4] Wang, Q., & Liu, H. (2023). Efficient Implementation of Sigmoid Activation Function Using Lookup Tables on FPGA. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 31(7), 1065-1073.

[5] Patel, R., & Gupta, S. (2024). Optimized Weight Quantization Techniques for FPGA-based Neural Network Inference. Proceedings of the International Conference on Field-Programmable Logic and Applications (FPLA), 245-252.

[6] Lee, C., & Kim, S. (2023). Accelerating Deep Learning Inference on FPGAs: A Survey. ACM Transactions on Reconfigurable Technology and Systems (TRETS), 10(2), 1-23.

[7] Zhang, L., & Chen, Y. (2023). FPGA-based Neural Network Acceleration: Challenges and Opportunities. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 42(9), 1803-1817.

[8] Tan, L., & Liu, Y. (2022). Real-time Handwritten Digit Recognition Using FPGA-accelerated Neural Networks. Journal of Embedded Systems, 17(4), 342-355.

[9] Wu, H., & Li, X. (2024). FPGA-based Acceleration of Convolutional Neural Networks for Edge Computing Applications. Proceedings of the International Symposium on Field-Programmable Gate Arrays (ISFPGA), 87-94.

[10] Garcia, M., & Martinez, E. (2023). Hardware-efficient Approximation of Activation Functions for FPGA-based Neural Network Inference. Journal of Low Power Electronics, 19(2), 212-225.

[11] Yang, Z., & Wang, X. (2022). A Comparative Study of FPGA-based Neural Network Accelerators. Proceedings of the International Conference on Field-Programmable Technology (ICFPT), 331-338.

[12] Huang, Q., & Zhao, L. (2024). FPGA-based Implementation of Deep Neural Networks for Real-time Image Classification. IEEE Embedded Systems Letters, 10(2), 41-48.

[13] Ali, Haitham & Mohammed, Esraa. (2010). Design Artificial Neural Network Using FPGA. International Journal of Computer Science and Network Security. 10.

[14] H - Feedforward Neural Network Implementation in FPGA Using Layer Multiplexing for Effective Resource Utilization