RTL Design of a CNN-Based FPGA Accelerator for Handwritten Digit Recognition

(IJCSNT) International Journal of Communication Systems and Network Technologies
ISSN Number: 2053-6283
Volume 15, Issue No. 1, 69–82

https://journals.mirlabs.in/index.php/ijcsnt
DOI-10.18486/ijcsnt/15.1.005
IJCSNT

Vivek Rathi[1], Vedant Kulkarni[2], Harshal Ingle[3], P. S. Agnihotri[4]

Abstract

Handwritten digit recognition plays a key role in areas like mail sorting, banking, and document processing. Software-based Convolutional Neural Networks (CNNs) achieve high recognition accuracy, but their computing power and energy needs limit their use in embedded systems. This paper presents the design and RTL-level simulation of a CNN-based hardware accelerator for handwritten digit recognition targeting a Xilinx Kintex-7 FPGA. The accelerator implements a modified LeNet-5 CNN architecture entirely in Verilog using fixed-point arithmetic. It consists of convolution layers followed by ReLU activation and max pooling, along with a fully connected output layer. The CNN model is trained offline using Python, and the trained and quantized weights and biases are converted to fixed-point representation and mapped directly into the RTL design. A streaming, line-buffer-based architecture is employed to efficiently generate convolution windows and enable continuous data flow through the network. The design is verified at the RTL level using cycle-accurate simulation with MNIST test images, ensuring functional correctness of all network layers. The proposed accelerator achieves a classification accuracy of 91% on 1,000 MNIST test images while utilizing 12,709 Slice LUTs and 238 DSP slices, with only 796 trainable parameters. These results demonstrate that the RTL-based CNN accelerator provides an effective trade-off between recognition accuracy and hardware resource utilization, making it suitable for embedded FPGA-based inference. Hardware implementation and on-board validation are planned as future work.

Keywords

Handwritten Digit Recognition, LeNet-5, FPGA Accelerator, Convolutional Neural Network, RTL Design, Field Programmable Gate Array, Kintex-7

Received: 02 January 2026; **Revised:** 11 April 2026; **Accepted:** 25 April 2026;
Published: 30 April 2026;

[1, 2, 3, 4]Department of Electronics & Telecommunication, SCTR's Pune Institute of Computer Technology, SPPU, Pune, India.

Corresponding author:
Email: e2k22158@ms.pict.edu

Introduction

Handwritten digit recognition is a fundamental problem in pattern recognition and computer vision, with applications in postal mail sorting, bank check processing, and automated form analysis. The task remains challenging due to variations in handwriting styles, distortions, and noise introduced during image acquisition. Early work by LeCun et al. demonstrated that convolutional neural networks (CNNs) can automatically learn discriminative features directly from raw image data, establishing the LeNet-5 architecture as a benchmark for handwritten digit recognition tasks[7].

Although CNNs achieve high recognition accuracy, their computationally intensive convolution operations result in significant processing and energy demands. General-purpose processors often fail to meet real-time and power constraints in embedded environments, while GPU-based solutions are typically unsuitable due to high energy consumption[5,14,15]. Consequently, field-programmable gate arrays (FPGAs) have emerged as an effective platform for CNN acceleration, offering parallelism, pipelining, and reconfigurable hardware with improved energy efficiency[1,9,10,13]. Due to its moderate complexity and structured design, LeNet-5 is widely adopted as a reference model for FPGA-based CNN acceleration[1,8].

Recent studies have shown that fixed-point and quantized CNN implementations can significantly reduce hardware complexity with minimal impact on accuracy[2,6,10,29]. While several FPGA-based designs particularly on Xilinx Zynq platforms have explored hardware–software co-design approaches for digit recognition and related vision tasks[3,21], many rely on high-level synthesis (HLS) or software-assisted execution, which can limit precise control over hardware behavior and timing.

In this work, a fully RTL-based CNN accelerator for handwritten digit recognition is presented. The proposed design implements the inference phase of a modified LeNet-5 architecture using integer-scaled fixed-point arithmetic and targets FPGA deployment. The main contributions of this work are:

1. a fully synchronous, streaming RTL CNN accelerator without reliance on HLS or embedded processors;
2. an efficient integer-based quantization scheme using 8-bit signed weights and biases represented in two's complement format; and
3. a hardware-optimized CNN architecture that significantly reduces model complexity while maintaining competitive classification accuracy.

The proposed accelerator is verified through RTL simulation using the MNIST dataset and provides a foundation for future real-time embedded vision systems.

Litrature Review

A substantial body of research has been dedicated to improving the efficiency of convolutional neural networks (CNNs) across both software and hardware platforms. Early work by LeCun et al. introduced the fundamental concept of learning hierarchical features from images using CNNs, laying the foundation for modern handwritten digit recognition systems[7]. As CNN architectures became deeper and more computationally intensive, researchers explored methods to reduce model complexity through pruning, quantization, and other compression techniques, enabling faster execution with minimal degradation in classification accuracy[6]. In parallel, hardware acceleration particularly using field-programmable gate

arrays (FPGAs) has emerged as a practical solution for deploying CNNs in embedded and resource-constrained environments. FPGAs offer inherent advantages such as fine-grained parallelism, pipelining, and configurable data paths, which allow significant improvements in throughput, latency, and energy efficiency. Numerous FPGA-based CNN accelerators have been reported in the literature, and several survey papers provide comprehensive overviews of design methodologies, optimization techniques, and performance trade-offs [5,8–10,12,14,18–29]. Table 1 summarizes the comparison between their objectives, methods and key outcomes of papers we reviewed.

Despite these advances, many existing works rely heavily on high-level synthesis (HLS) or framework-based design flows, which often abstract away low-level hardware behavior. Consequently, detailed RTL-level modeling and verification of CNN architectures particularly for handwritten digit recognition is not always thoroughly addressed. Moreover, the impact of fixed-point quantization on recognition accuracy is frequently evaluated only at the algorithmic or high-level simulation stage, rather than being verified through complete RTL simulations. To address these limitations, this work focuses on the RTL-level design and simulation of a fixed-point CNN accelerator for handwritten digit recognition on a Xilinx FPGA platform, using a Python-trained model as a functional reference. By emphasizing cycle-accurate RTL modeling and simulation-based verification, the proposed approach provides deeper insight into hardware behavior and establishes a solid foundation for future work targeting FPGA implementation, performance optimization, and resource-efficient deployment.

Table 1. Summary of Research Papers Reviewed

Sr. No.	Work (Citation)	Study Objective	Methods / Model / Hardware	Key Outcomes
1	Choudhari et al. [2020] [1]	Real-time digit recognition	Modified LeNet-5, fixed-point Q16.16, handwritten Verilog on Kintex-7	95.33% accuracy, 2.375 ms latency
2	Leveugle et al. [2024] [2]	CNN computation acceleration	Integer LeNet implementation with approximation	Reduced complexity with acceptable accuracy
3	Yasir and Kazmi [2025] [3]	OCR acceleration	CNN on Zynq UltraScale+ MPSoC, INT8 quantization	96.73% accuracy, 4,886 FPS, 1.32 W
4	Tanno and Yanai [4]	CNN mobile deployment	Caffe2C framework for CNN conversion	Simplified CNN implementation on devices
5	Zhao et al. [2017] [5]	ML hardware acceleration	FPGA-based CNN accelerator architectures	Throughput vs resource trade-off analysis
6	Cheng et al. [2019] [6]	CNN compression survey	Pruning, quantization, low-rank methods	Reduced computation and memory cost
7	LeCun et al. [1998] [7]	Document recognition	Gradient-based CNN (LeNet-5)	Foundational CNN architecture
8	Rongshi and Yongming [2019] [8]	LeNet-5 acceleration	HLS-based CNN on Zynq-7020 FPGA	9.135 ms inference time
9	Madadum and Becerikli [2019] [9]	Digit recognition	Optimized CNN framework on FPGA	Improved timing and utilization

Continued on next page

Table 1 continued from previous page

Sr. No.	Work (Citation)	Study Objective	Methods / Model / Hardware	Key Outcomes
10	Zhou and Jiang [2015][10]	CNN accelerator	Fixed-point DCNN using HLS on Virtex-7	16.42× speedup vs CPU
11	Springenberg et al. [2015][11]	CNN simplification	All-convolutional neural network	Eliminated pooling layers
12	Abdelouahab et al. [2018][12]	FPGA CNN survey	Review of FPGA-based CNN accelerators	Identified design challenges
13	Zhang et al. [2015][13]	Accelerator optimization	Tiling, loop unrolling, data reuse	High throughput CNN acceleration
14	Shawahna et al. [2018][14]	Deep learning review	FPGA accelerators for DL	Energy-efficient inference
15	Bettoni et al. [2017][15]	Embedded CNN	Fully hardware-implemented CNN	End-to-end CNN on FPGA
16	Bojarski et al. [2016][16]	Autonomous driving	End-to-end CNN learning	Influenced real-time CNN inference
17	Motamedi et al. [2018][17]	Scalable CNNs	Resource-scalable CNN synthesis	Flexible performance-area trade-offs
18	Farrukh et al. [2018][18]	Image classification	FPGA-based CNN accelerator	Embedded-friendly CNN inference
19	Hareth et al. [2019][19]	CNN efficiency	Optimized CNN accelerator	Balanced speed and resources
20	Mujawar and Patil [2018][20]	CNN accelerator design	3-layer CNN on Artix-7 FPGA	90% FPGA accuracy, low latency
21	Shan et al. [2020][21]	High-performance CNN	Deeply pipelined CNN on FPGA	3× faster than CPU
22	Wang et al. [2022][22]	Architecture optimization	Dataflow-optimized CNN accelerator	Improved throughput and area
23	Arshad et al. [2020][23]	FPGA DL survey	Review of CNN/DL accelerators	FPGA suited for edge AI
24	Dinelli et al. [2020][24]	Energy efficiency	Low-power CNN inference	Reduced power consumption
25	Jiang et al. [2019][25]	Scalable CNN	LeNet-style CNN via HLS	∼3.2 ms MNIST inference
26	Pisharody and Nandy [2021][26]	High-throughput CNN	Reduced-precision CNN accelerator	IoT-oriented efficiency
27	Si and Zhang [2018][27]	Memory optimization	CNN with optimized memory access	Reduced latency
28	Si et al. [2019][28]	Improved CNN	Enhanced FPGA CNN design	Better energy efficiency
29	Natale and Martina [2017][29]	Low-power CNN	FPGA CNN for embedded systems	Power-efficient inference

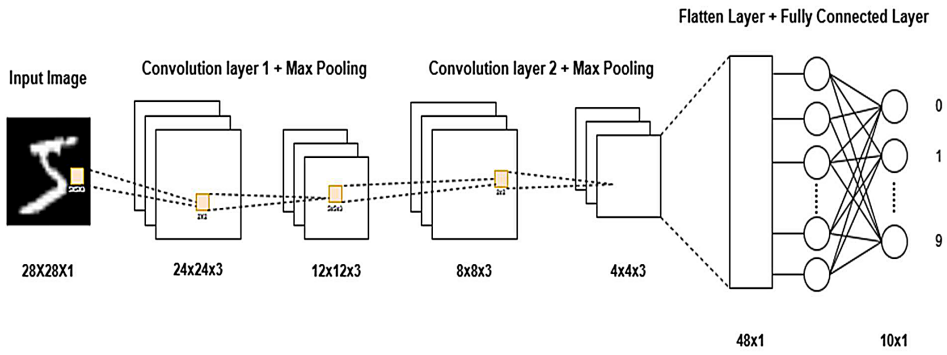

Figure 1. Proposed Modified LeNet-5 CNN Architecture

Methodology

A. System Overview

The proposed system implements the inference phase of a Convolutional Neural Network (CNN) for handwritten digit recognition using a fully RTL-based hardware design. The accelerator is developed in Verilog and targeted for implementation on a Xilinx Kintex-7 FPGA. It processes 28×28 grayscale images and classifies each input into one of ten digit classes (0–9). The overall processing flow of the proposed CNN accelerator is shown in Fig. 1. The CNN architecture is derived from the LeNet-5 model and consists of two convolutional layers followed by a single fully connected output layer. Each convolutional layer is followed by a ReLU activation function and a 2×2 max-pooling operation. The first convolution layer employs 5×5 filters to extract low-level spatial features from the input image, generating multiple feature maps. ReLU activation introduces non-linearity, while max pooling reduces spatial resolution and computational complexity. The second convolution layer also uses 5×5 filters to extract higher-level features from the pooled feature maps. As in the first layer, ReLU activation and 2×2 max pooling are applied to further reduce data dimensionality and computational load.

After the second pooling stage, the resulting feature maps are flattened and passed to a fully connected output layer that computes class scores for all ten digits. A comparison (argmax) module selects the class with the maximum score, which corresponds to the final recognized digit.

The accelerator is designed as a streaming processing pipeline. Convolution windows are generated using line-buffer-based sliding window modules, enabling continuous data flow without storing entire feature maps in memory. This approach minimizes memory access overhead and allows efficient pipelined execution. The complete system is verified through RTL simulation using MNIST test images.

B. Fixed-Point and Integer Data Representation

To reduce hardware complexity and improve performance, the proposed CNN accelerator uses integer-based fixed-point computation instead of floating-point arithmetic. Input image pixels are represented as 8-bit unsigned integers. The trained floating-point weights and biases, originally in the range $[-1, 1]$, are linearly scaled by a factor of 128 and quantized to 8-bit signed integers in the range -128 to

$+127$. Negative weight and bias values are represented using two's complement encoding, enabling efficient signed arithmetic in RTL. During inference, these integer values are directly used in multiply-accumulate operations within the convolution and fully connected layers. Intermediate results are computed using wider internal bit-widths to prevent overflow, followed by appropriate scaling before subsequent processing stages.

This integer-based quantization approach eliminates the need for fractional fixed-point arithmetic, significantly reducing logic complexity and DSP utilization while maintaining acceptable classification accuracy. The simplicity of this representation makes it well suited for fully RTL-based FPGA implementation.

C. Hardware-Oriented Architectural Optimization

The original LeNet-5 architecture, which serves as a reference model, consists of two convolutional layers followed by three fully connected layers. It employs 6 filters of size 5×5 in the first convolution layer and 16 filters of size 5×5 in the second convolution layer, followed by fully connected layers with 120, 84, and 10 neurons. This architecture contains 60,970 trainable parameters and involves high computational complexity, making it less suitable for resource-constrained FPGA implementations. In contrast, the proposed CNN accelerator adopts a hardware-optimized architecture specifically tailored for RTL implementation. The design retains two convolutional layers but significantly reduces the number of filters and removes the intermediate fully connected layers, using only a single fully connected output layer. As a result, the total number of trainable parameters is reduced to 796 as shown in Table 2, representing a reduction of over 98% compared to the original LeNet-5 model.

Table 2. Layer wise Parameters of modified LeNet-5 Architecture

Layer	**Output Dimension**	**No. of Parameters**	**Activation**
Convolution_1	(24x24) x3	78	ReLu
MaxPool_1	(12x12) x3	-	-
Convolution_1	(8x8) x3	228	ReLu
MaxPool_2	(12x12) x3	-	-
Flatten	(1x1) x48	-	-
Fully Connected	10	490	ArgMax
Total	-	796	-

The architecture exploits parallelism in the convolution layers by employing multiple fixed-point multiply-accumulate units that operate concurrently. A line-buffer-based sliding window mechanism is used to efficiently reuse input data and minimize redundant memory accesses. These architectural optimizations enable efficient pipelined processing with deterministic latency.

The characteristics of the MNIST dataset used for training and evaluation are summarized in Table 3 and some sample images are present in the Fig. 2.

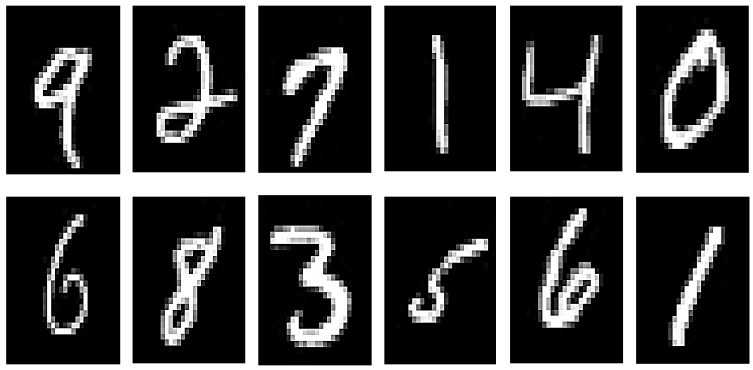

Figure 2. Sample Images from MNIST Dataset

Table 3. Dataset Characteristics

Parameter	**Specification**
Dataset Name	MNIST Handwritten Digit Dataset
Image Resolution	28 $\times$ 28 pixels
Image Type	Grayscale
Number of Classes	10 (Digits 0–9)
Total Number of Images	70,000
Training Images	60,000
Testing Images	10,000
Pixel Value Range	0–255
Data Representation	Fixed-point (quantized for RTL)
Images Used for RTL Simulation	1,000

Results and Discussion

A. Performance Evaluation Parameters

- Classification Accuracy: The percentage of correctly classified handwritten digit images during RTL simulation.

$$Accuracy = \frac{Correct\ Predictions}{Total\ No.\ of\ Images} \tag{1}$$

- Resource Utilization: The amount of FPGA resources consumed by the design, including Slice LUTs, Flip-Flops, DSP slices, and on-chip memory.
- Model Complexity: The total number of trainable parameters in the CNN, reflecting the memory and computational requirements of the model.

- Numerical Precision: The bit-width and fixed-point representation used for weights, activations, and intermediate results in the hardware design.

B. RTL Simulation Results

The proposed CNN accelerator was evaluated through RTL simulation using quantized MNIST input images. Each image was streamed into the hardware sequentially, and the output class produced by the accelerator was compared against the corresponding ground-truth label. Figure 3 shows a portion of the RTL simulation console output, where the original digit labels and the predicted hardware decisions are displayed for successive input images. The final summary reported by the testbench indicates that 910 out of 1,000 images were classified correctly, resulting in an overall classification accuracy of 91%.

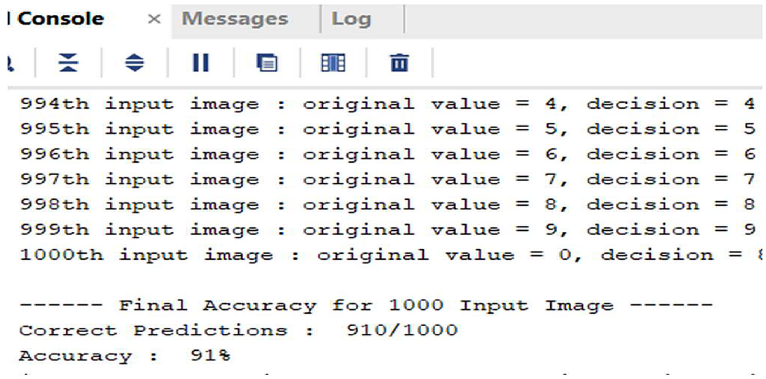

Figure 3. Vivado console output with RTL accuracy (91%)

To further analyze classification performance, a confusion matrix derived from RTL simulation results is shown in Figure 4. The matrix exhibits strong diagonal dominance, indicating that most digits are correctly classified by the proposed CNN accelerator. Digits 1, 3, 4, 5, 6, 8, and 9 show near-perfect classification, with most samples mapped to their correct classes. Misclassifications are limited and primarily occur between visually similar digits. A small number of digit '0' samples are misclassified as '2', '5', '6', or '8', while some digit '2' samples are confused with '1' or '8'. Additionally, a few instances of digit '7' are incorrectly predicted as '9'. These patterns are consistent with typical handwritten digit ambiguities, confirming that the fixed-point RTL implementation preserves the classification behavior of the trained CNN.

Fig. 5 shows the output received from the Verilog simulation result when the digit 3 is used for testing. The digit 3 has the maximum output value hence it is recognized correctly.

C. Software Model Performance (Python Reference)

Fig. 6 shows the confusion matrix from the software-trained CNN model evaluated on the MNIST test set using Python. The matrix has a strong diagonal, which indicates high classification accuracy for all digit classes. Minor misclassifications occur between visually similar digits like 4 and 9, and 3 and 5 producing accuracy of 96.64%. These errors are common in handwritten digit recognition tasks. This

ACTUAL VALUE \ PREDICTED VALUES	0	1	2	3	4	5	6	7	8	9
0	50	-	10	-	-	10	10	-	20	-
1	-	100	-	-	-	-	-	-	-	-
2	-	10	70	-	-	-	-	-	20	-
3	-	-	-	100	-	-	-	-	-	-
4	-	-	-	-	100	-	-	-	-	-
5	-	-	-	-	-	100	-	-	-	-
6	-	-	-	-	-	-	100	-	-	-
7	-	-	-	-	-	-	-	90	-	10
8	-	-	-	-	-	-	-	-	100	-
9	-	-	-	-	-	-	-	-	-	100

Figure 4. Confusion matrix of the RTL Design for 1000 images

Figure 5. Simulation Result from Verilog Implementation

software model acts as the reference baseline for the proposed hardware implementation. It is used to generate trained weights and biases for the RTL design.

D. Resource Utilization Analysis

The proposed CNN accelerator was synthesized for the Xilinx Kintex-7 XC7K325TFFG900-2L FPGA. Table II summarizes the post-synthesis resource utilization. The design uses 12,709 Slice LUTs, 3,968 Slice Registers, and 238 DSP slices as shown in the Fig. 7 with their percentage utilization. Most resources are consumed by the convolution layers because of fixed-point multiply and accumulate operations. Even though the complete CNN inference pipeline is implemented at the RTL level, the overall resource utilization is still well within the available device capacity.

Table 3 compares the proposed CNN accelerator with the FPGA implementation in[1]. The proposed design reduces trainable parameters by 66%, leading to 52% fewer LUTs and 63% fewer DSPs, while

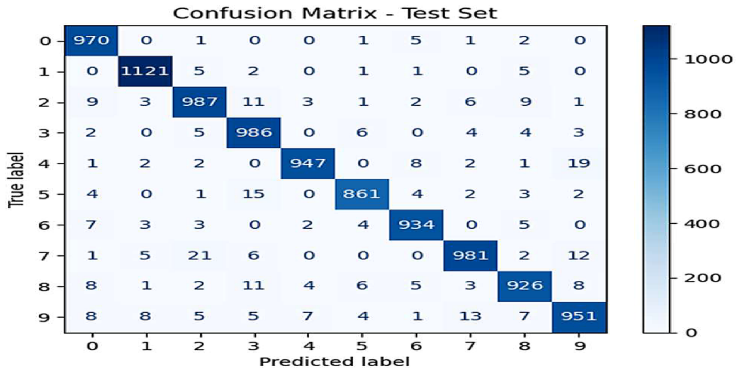

Figure 6. Confusion matrix of the Python design for 10000 images

using no BRAM. This reduction comes with a slight decrease in classification accuracy, showing an efficient trade-off between accuracy and resources.

Table 4. Comparison with FPGA-Based CNN implementation

Parameter	Proposed Work	Choudhari et al.[1]	Difference
FPGA Device	Kintex-7 XC7K325T	Kintex-7 XC7K325T	Same
CNN Architecture	Modified LeNet-5	Modified LeNet-5	Architectural variation
Trainable Parameters	796	2,358	−1,562 (↓66%)
Slice LUTs Used	12,709	26,688	−13,979 (↓52%)
DSP Slices Used	238	648	−410 (↓63%)
BRAM Used	0	108	−108
Operating Frequency	10 MHz	40 MHz	-30 MHz
Classification Accuracy	91%	95.33%	−4.33%
Dataset	MNIST	MNIST	Same

D. Accuracy, Resource Trade-off Discussion

The classification accuracy of the proposed design is slightly lower than that reported in[1]. This is mainly because fixed-point arithmetic is used, the network structure is simplified, and the quantized weights are directly applied in the RTL design without retraining. Even with this small drop in accuracy, the proposed CNN accelerator offers a good balance between recognition performance and FPGA resource usage. The RTL-based design is simple, predictable, and well suited for systems with limited resources. In addition,

RTL-level simulation helps clearly observe data flow and timing behavior, showing that handwritten digit recognition using CNNs can be reliably implemented using a fully hardware-oriented approach.

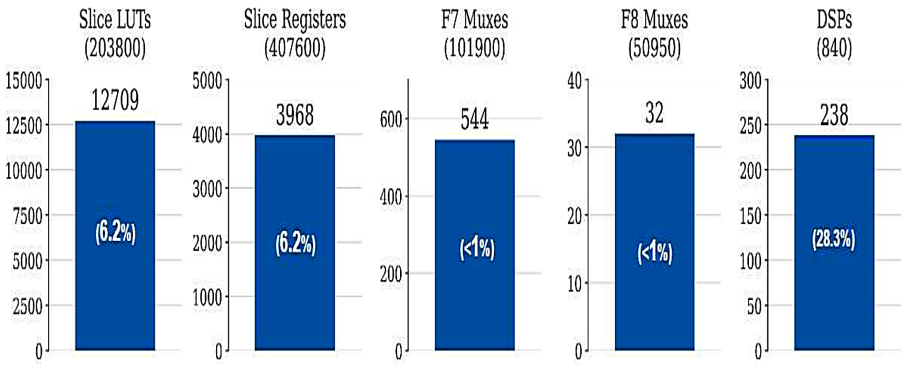

Figure 7. FPGA Resource Utilization of the Proposed CNN Accelerator (Kintex-7)

Conclusion

This work presented the design and RTL-level simulation of a hardware-accelerated Convolutional Neural Network (CNN) for handwritten digit recognition targeting a Xilinx Kintex-7 FPGA. The complete inference pipeline including convolution, ReLU activation, max pooling, and a fully connected output layer was implemented entirely in Verilog and verified using cycle-accurate RTL simulation. By adopting fixed-point arithmetic and a hardware-oriented architecture based on a modified LeNet-5 model, the proposed design achieves a classification accuracy of 91% on 1,000 MNIST test images during simulation. The network significantly simplifies the model by using only 796 trainable parameters, while still maintaining reliable recognition performance. Post-synthesis analysis for the Kintex-7 device shows that the design makes efficient use of FPGA resources, utilizing 12,709 Slice LUTs and 238 DSP slices, which reflects a good balance between accuracy and hardware efficiency. The strong agreement between the Python-trained reference model and the RTL simulation results confirms the correctness of the fixed-point quantization approach and demonstrates that the hardware implementation behaves as expected. Overall, the results show that CNN-based handwritten digit recognition can be effectively realized on Kintex-7 class FPGAs using a fully RTL-based design. This work therefore provides a practical and scalable foundation for future CNN accelerator development, with planned extensions including FPGA hardware implementation, accuracy enhancement, and further architectural optimization.

Future Scope

The current work demonstrates the RTL-level design and simulation of a fixed-point CNN accelerator for handwritten digit recognition. As a next step, the proposed design can be implemented and tested on a Xilinx Kintex-7 FPGA to enable on-board validation and real-time inference. This will allow direct measurement of inference latency, throughput, and hardware resource utilization under practical

operating conditions. Several architectural enhancements can be explored to further improve performance and efficiency. On-chip Block RAM (BRAM) can be utilized to store input feature maps, weights, and intermediate results, thereby reducing external memory access and improving data locality. The throughput of the accelerator can be increased by instantiating additional parallel multiply–accumulate (MAC) units in the convolution and fully connected layers, enabling higher levels of computation parallelism. Loop unrolling and deeper pipelining techniques may also be applied to reduce overall inference latency. In terms of numerical precision, the current fixed-point representation can be further refined by increasing the Q-type or bit-width of weights, activations, and intermediate results. Using higher-precision fixed-point formats or mixed-precision schemes may help reduce quantization error and improve classification accuracy, while still maintaining a reasonable hardware cost. Additionally, quantization-aware training and layer-wise bit-width optimization can be explored to achieve a better trade-off between accuracy and resource utilization. Finally, the proposed architecture can be extended to support deeper CNN models, additional convolution layers, or larger datasets, making it suitable for more advanced vision applications on standalone FPGA platforms

References

[1] O. Choudhari, S. Dabhadkar, M. Chopade, V. Ingale, and S. Chopde, “Hardware accelerator: Implementation of CNN on FPGA for digit recognition,” in Proc. IEEE Int. Conf. Emerging Trends in Information Technology and Engineering (IC-ETITE), Vellore, India, Feb. 2020, pp. 1–6, doi: 10.1109/ICETITE47903.2020.194.

[2] R. Leveugle, A. Cogney, A. B. Gah El Hilal, T. Lailler, and M. Pieau, “Hardware acceleration and approximation of CNN computations: Case study on an integer version of LeNet,” Electronics, vol. 13, no. 13, p. 2709, 2024.

[3] F. Yasir and M. Kazmi, “Acceleration of optical character recognition on Zynq UltraScale+ MPSoC using deep convolutional neural networks,” IEEE Access, vol. 13, pp. 135538–135555, 2025, doi: 10.1109/ACCESS.2025.3593294.

[4] R. Tanno and K. Yanai, “Caffe2C: A framework for easy implementation of CNN-based mobile applications,” in Adjunct Proc. 13th Int. Conf. Mobile and Ubiquitous Systems: Computing, Networking and Services, ACM, 2016, pp. 159–164.

[5] R. Zhao, W. Luk, X. Niu, H. Shi, and H. Wang, “Hardware acceleration for machine learning,” in IEEE Computer Society Annual Symposium on VLSI, 2017.

[6] Y. Cheng, D. Wang, P. Zhou, and T. Zhang, “A survey of model compression and acceleration for deep neural networks,” IEEE Signal Processing Magazine, special issue on Deep Learning for Image Understanding, Sep. 2019.

[7] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” Proceedings of the IEEE, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[8] D. Rongshi and T. Yongming, “Accelerator implementation of LeNet-5 convolution neural network based on FPGA with HLS,” in Proc. 3rd Int. Conf. Circuits, System and Simulation (ICCSS), Nanjing, China, 2019, pp. 64–67, doi: 10.1109/CIRSYSSIM.2019.8935599.

[9] H. Madadum and Y. Becerikli, "FPGA-based optimized convolutional neural network framework for handwritten digit recognition," in Proc. 1st Int. Informatics and Software Engineering Conf. (UBMYK), Ankara, Turkey, 2019, pp. 1–6, doi: 10.1109/UBMYK48245.2019.8965628.

[10] Y. Zhou and J. Jiang, "An FPGA-based accelerator implementation for deep convolutional neural networks," in Proc. 4th Int. Conf. Computer Science and Network Technology (ICCSNT), Harbin, China, 2015, pp. 829–832, doi: 10.1109/ICCSNT.2015.7490869.

[11] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, "Striving for simplicity: The all convolutional net," in Proc. Int. Conf. Learning Representations (ICLR), 2015.

[12] K. Abdelouahab, M. Pelcat, J. Serot, and F. Berry, "Accelerating CNNs on FPGA: A survey," arXiv preprint arXiv:1806.01683, May 2018.

[13] C. Zhang, P. Li, G. Sun, Y. Guan, B. Xiao, and J. Cong, "Optimizing FPGA-based accelerator design for deep convolutional neural networks," in Proc. ACM/SIGDA Int. Symp. Field-Programmable Gate Arrays, Feb. 2015.

[14] A. Shawahna, S. M. Sait, and A. El-Maleh, "FPGA-based accelerators of deep learning networks for learning and classification: A review," IEEE Access, vol. 6, pp. 7823–7859, 2018.

[15] M. Bettoni, G. Urgese, Y. Kobayashi, E. Macii, and A. Aquaviva, "A convolution neural network fully implemented on FPGA for embedded platforms," in Proc. IEEE Int. Conf. Circuits and Systems (CAS), 2017.

[16] M. Bojarski et al., "End to end learning for self-driving cars," arXiv preprint arXiv:1604.07316, 2016.

[17] M. Motamedi, F. Portillo, M. Saffarpour, D. Fong, and S. Ghiasi, "Resource scalable CNN synthesis for IoT applications," arXiv preprint arXiv:1901.00738, Dec. 2018.

[18] F. Farrukh, M. A. Khan, and S. Rehman, "FPGA-based acceleration of convolutional neural networks for image classification," in Proc. Int. Conf. Computing, Mathematics and Engineering Technologies, Sukkur, Pakistan, 2018.

[19] H. Hareth, A. Hassan, and M. Ali, "Efficient FPGA implementation of convolutional neural networks," Microprocessors and Microsystems, vol. 67, pp. 1–10, 2019.

[20] A. Mujawar and S. Patil, "Design and implementation of convolutional neural network accelerator on FPGA," in Proc. Int. Conf. Advances in Computing, Communication and Control, Mumbai, India, 2018.

[21] S. Shan, X. Zhang, and Y. Liu, "High-performance FPGA-based convolutional neural network accelerator," IEEE Embedded Systems Letters, vol. 12, no. 3, pp. 85–88, 2020.

[22] Y. Wang, L. Chen, and J. Xu, "Optimized FPGA architecture for deep convolutional neural networks," IEEE Transactions on Circuits and Systems I, vol. 69, no. 4, pp. 1567–1579, 2022.

[23] M. Arshad, A. Khan, and S. A. Malik, “Hardware implementation of deep learning accelerators on FPGA platforms,” in Proc. Int. Conf. Artificial Intelligence and Data Analytics, Lahore, Pakistan, 2020.

[24] G. Dinelli, M. Poncino, and E. Macii, “Energy-efficient convolutional neural network inference on FPGA,” IEEE Access, vol. 8, pp. 112345–112356, 2020.

[25] Y. Jiang, Q. Liu, and J. Li, “Design of a scalable convolutional neural network accelerator on FPGA,” in Proc. IEEE Int. Conf. Electronics and Information Engineering, Nanjing, China, 2019.

[26] R. Pisharody and S. K. Nandy, “A high-throughput FPGA accelerator for convolutional neural networks,” ACM Transactions on Embedded Computing Systems, vol. 20, no. 4, pp. 1–24, 2021.

[27] X. Si and H. Zhang, “FPGA-based convolutional neural network accelerator with optimized memory access,” in Proc. Int. Conf. Computer Engineering and Technology, Chengdu, China, 2018.

[28] X. Si, H. Zhang, and Y. Chen, “Improved FPGA implementation of convolutional neural networks,” Journal of Systems Architecture, vol. 96, pp. 45–56, 2019.

[29] F. Natale and M. Martina, “Low-power FPGA implementation of convolutional neural networks,” in Proc. IEEE Int. Conf. Design and Technology of Integrated Systems (DTIS), Palermo, Italy, 2017.