IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

FPGA-Based Object Detection Using Yolov8 With Verilog Implementation

K. B. L. Phani Kumar[1] Dr. K. Jhansi Rani[2]

[1]M. Tech, VLSI & EMBEDDED SYSTEM, ECE, UCEK, JNTU Kakinada, Andhra Pradesh, India
[2]Assistant Professor, Dept. Of ECE, UCEK, JNTU Kakinada, Andhra Pradesh, India.

ABSTRACT:

An object detection processor based on a binarized neural network (BNN) architecture is developed to enable efficient deployment on resource-constrained FPGA platforms. Traditional convolutional neural networks (CNNs), while highly effective for visual recognition tasks, impose significant demands on computational resources and memory bandwidth, present challenges for embedded applications. To address these limitations, the proposed system leverages binary operations, on-chip memory usage, and an optimized neural architecture to minimize complexity and power consumption. The design achieves accurate object detection performance within the limitations of FPGA hardware, highlighting its potential for efficient inference in real-time environments. Its modularity and scalability allow for adaptation across a wide range of embedded vision scenarios, particularly in domains requiring low latency and high energy efficiency. The proposed approach demonstrates the feasibility of integrating deep learning models into compact, resource-effective hardware systems, offering a promising direction for future applications in areas such as IoT monitoring, smart sensors, and mobile robotics.

Keywords:
Object Detection, FPGA, Binarized Neural Network, Embedded Vision, Resource Optimization, Low-Power Design, IoT, Robotics, Hardware Acceleration.

1. Introduction

The domain of computer vision has been transformed by the use of deep learning, a form of artificial intelligence that uses multi-layered neural networks to recognize and comprehend complex patterns from visual data. The technology has enabled the deployment of a wide variety of applications, including autonomous image processing and real-time decision-making systems, revolutionizing the manner in which machines perceive and interact with the visual environment. Deep learning algorithms, particularly those based on convolutional neural networks (CNNs), have emerged as powerful tools as they are able to process visual information hierarchically, approximating human vision features with enhanced accuracy. Pioneering papers in the area [3], [4] have established solid foundations that continue to evolve, driving innovation in areas [5] and motivating further research in efficient implementations [6]. Ongoing optimization of the algorithms has also driven interest in porting them to other hardware platforms, making them more accessible and implementable in practice.

1.1 The Significance of Object Detection

Object detection, which allows for the simultaneous identification and localization of multiple objects within images or video sequences, is a key component of deep learning applications. This ability is essential for developing technologies like security systems that keep an eye out for possible threats [3], autonomous cars that can detect obstacles to ensure safety [4], and robotic platforms that need environmental awareness to navigate and perform tasks [5]. Traditional feature-based approaches to object detection have given way to advanced CNN-driven techniques [6], which provide improved accuracy and flexibility in a variety of scenarios. Though this comes with high computational demands that test current hardware limitations, these developments have paved the way for the deployment of such systems in real-time scenarios where quick processing is crucial. The necessity for scalable and effective solutions to satisfy a variety of operational requirements is highlighted by the increasing dependence on object detection in common place technologies.

1.2 Embedded Systems Difficulties

Although object detection has advanced, there are still significant obstacles to overcome when implementing CNN-based solutions on embedded and mobile platforms. Large parameter sets and complex operations are hallmarks of these model's complex architectures, which require significant memory and processing power that are frequently beyond the capabilities of hardware with limited resources. Traditional CNN deployment is a difficult task because field-programmable gate arrays (FPGAs), which are frequently used in low-power or cost-sensitive applications, have limitations in logic elements and on-chip storage. Power consumption and latency problems are made worse by the dependence on high-precision calculations and frequent external memory access. These problems are especially important in devices that run on batteries or have limited resources. To enable the realistic integration of advanced vision tasks into embedded systems, this has prompted the investigation of alternative architectures and optimization techniques, leading to a new wave of research centred on efficiency and adaptability [1].

1.3 Background of the study

The work by Lee et al. [1], which presented a real-time object detection processor based on binarized neural networks (BNNs), is a famous step toward resolving these issues. This study, which was published in a prestigious journal, suggests a method for algorithm-hardware co-optimization that uses XNOR-based CPUs and a unique layer structure to lessen the information loss that comes with deep quantization. The design shows that BNNs can be used for real-time applications by lowering computational overhead through bitwise operations, providing a good substitute for conventional floating-point calculations. It is implemented on a high-end FPGA platform. By utilizing on-chip resources, this method does away with the need for off-chip memory and establishes a standard for later adaptations to less powerful hardware. Because of this framework's success, more research into its scalability has been conducted, leading to attempts to adapt it to a wider variety of devices and applications [1].

This work redefines the design by Lee et al. [1] for the FPGA, a platform with limited hardware resources, driven by the need to extend such capabilities to resource-constrained environments. In order to accommodate the platform's limitations and ensure compatibility with its limited architecture, the project modifies the BNN framework by using an alternative data interface and a programmable array of XNOR-based processing elements. By using on-chip storage and a simplified model structure to match the available resources, the architecture is made to function without the need for external memory. Validated by synthesis and testing tools to evaluate performance and dependability, initial implementation and subsequent optimizations explore improvements in data transfer efficiency and processing configuration. With the help of cutting-edge interfaces and hardware advancements, this attempt seeks to provide a

scalable solution for embedded vision applications, including low-power robotics and Internet of Things surveillance. In order to provide a thorough examination of this effort to develop embedded vision technology and support the continuous development of low-cost intelligent systems, the paper goes on to describe the relevant work, suggested methodology, implementation process, and comparative analysis.

2. Associated Research

Recent advances in deep learning have significantly improved the performance of computer vision systems, particularly in tasks such as object detection and classification. Convolutional Neural Networks (CNNs) form the basis of many state-of-the-art models due to their ability to capture spatial hierarchies in image data. However, their high computational and memory demands present challenges for deployment in low-power embedded systems.

To address this, researchers have proposed various model optimization techniques. Among them, Binarized Neural Networks (BNNs) have gained attention for their ability to reduce both memory usage and computational complexity. Notable works such as XNOR-Net [7] introduced binary operations—replacing floating-point multiplications with XNOR and bit-counting—to significantly accelerate inference. Bi-Real Net [8] further improved training stability and accuracy by integrating shortcut connections into binary models, while Real-to-Binary Networks [9] added scaling factors for better approximation of real-valued representations. These architectures have shown promise in enabling efficient deep learning on constrained hardware platforms, such as FPGAs and ASICs.

Object detection models have evolved rapidly with the development of single-shot detectors, which enable high-speed inference. The YOLO (You Only Look Once) family of models exemplifies this evolution by offering real-time object detection with competitive accuracy. Earlier versions focused on lightweight design, while recent iterations like YOLOv8 by Ultralytics incorporate enhancements in backbone networks, feature fusion, and output layers.

YOLOv8 typically employs a backbone such as CSPDarknet, a neck based on Feature Pyramid Networks (FPN), and a detection head that outputs bounding boxes and class scores. These improvements allow it to handle objects of various scales effectively. The model has been widely adopted for real-time applications in robotics, surveillance, and edge computing, and is often benchmarked using datasets like Pascal VOC 2007+2012, which offer diverse object classes and scene variations. Figure 1 presents the architecture of the YOLOv8 model, which consists of three main stages: the CSPDarknet-based backbone for feature extraction, an FPN-based neck for multi-scale fusion, and a detection head for object classification and localization.

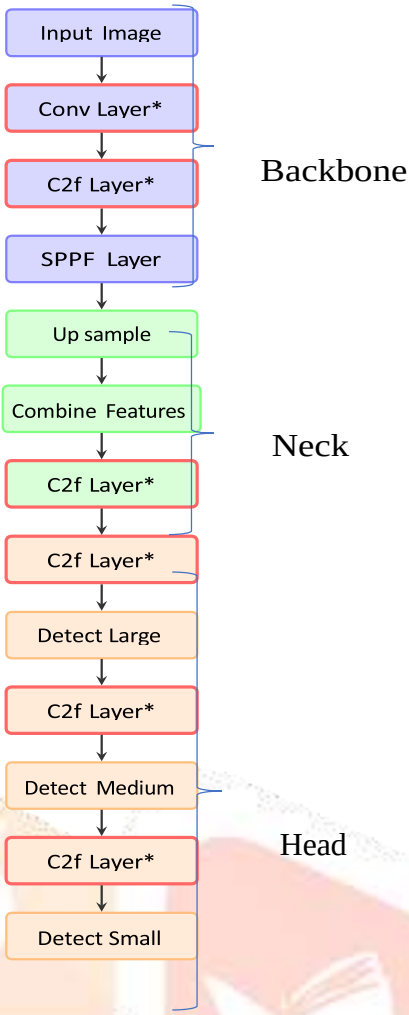

Figure. 1: YOLOv8 Architecture Adapted for BNN Implementation.

The use of hardware description languages such as Verilog has enabled efficient deployment of neural networks on reconfigurable platforms like FPGAs. This approach facilitates the implementation of custom digital circuits optimized for parallelism and reduced latency.

Early Verilog-based neural network accelerators primarily focused on standard CNNs. However, recent work has shifted toward BNN-specific architectures, where conventional multiply-accumulate units are replaced with logic-based XNOR operations. For instance, Lee et al. [1] proposed a variable-precision processing unit tailored for BNNs, demonstrating substantial reductions in power and memory requirements. These efforts underscore the importance of hardware-software co-design, particularly when adapting deep learning models to FPGA constraints such as limited logic elements and on-chip memory. The structure and operation of the binary processing element (PE) array used for convolution are illustrated in Figure 2. It shows the arrangement of XNOR engines, BRAM blocks, and control logic necessary for binary computation.

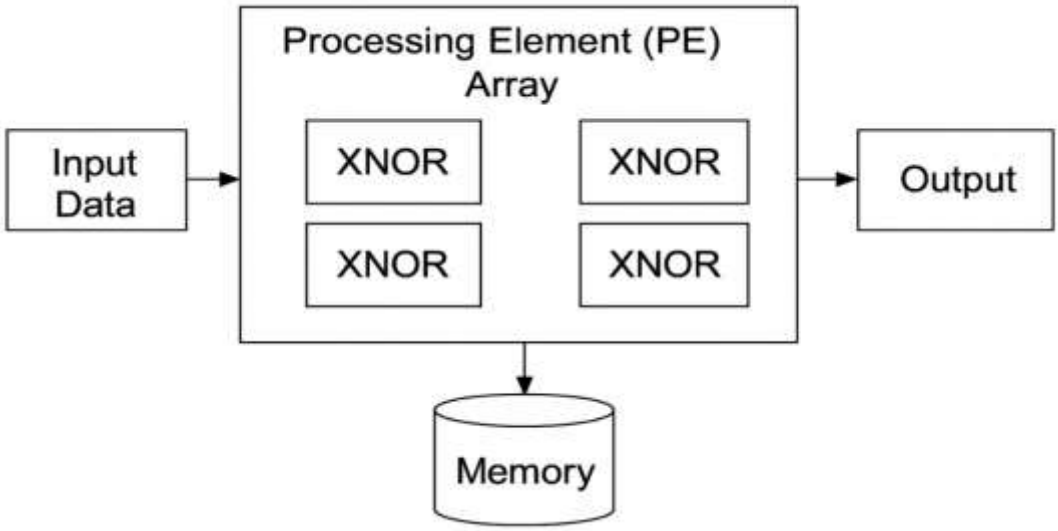

Figure. 2: Block Diagram of Verilog-Based Processing Element Array.

FPGAs have emerged as a suitable platform for real-time deep learning inference due to their inherent parallelism and flexibility. Several efforts have focused on adapting object detection models for FPGA deployment. For example, Lightweight YOLOv2 [10] and Sim-YOLO-v2 [11] explored hardware-aware redesigns of YOLO variants, often relying on external memory to handle large model parameters. FPGA-based object detection using YOLO architectures has become a promising approach for real-time embedded vision in IoT, robotics, and autonomous systems. Research highlights that quantization techniques (binary, multi-bit, and fixed-point) along with hardware-level optimizations such as pipelining and memory reuse are essential to balance accuracy, throughput, and resource efficiency. While binary neural networks offer significant reductions in computation and memory cost, they often compromise detection accuracy, making mixed-precision strategies more effective for advanced models like YOLOv8. Overall, FPGA deployment of YOLO enables low-power, real-time vision systems, though challenges remain in handling complex backbones and minimizing external memory dependence.[2]

Other research, such as Real-Time SSDLite [14], focused on improving inference throughput on embedded devices, while Tincy YOLO [12] and Code Net [13] proposed architectural compression and deformable convolution strategies for efficient detection. Lee et al. [1] demonstrated an FPGA-based BNN processor that avoided external memory usage by depending entirely on on-chip storage, showcasing a viable method for power-efficient object detection in constrained environments. These implementations highlight the ongoing interest in achieving low-latency and energy-efficient inference through FPGA-based solutions.

A major trend in embedded AI research is the search of energy- and resource-efficient architectures that maintain acceptable levels of accuracy. This includes lightweight models, aggressive quantization, pruning, and the adoption of BNNs. Research derived from XNOR-Net principles has led to hardware-friendly models capable of running on low-end FPGAs with constrained resources.

Several works have investigated simplified data interfaces and processing element arrays optimized for small devices, often scaling down high-performance designs for cost-sensitive applications. By emphasizing on-chip computation, minimal external memory usage, and architectural simplification, these solutions align with the growing demand for deployable deep learning on edge devices, particularly in sectors such as IoT, surveillance, and autonomous systems.

The collective body of work in model optimization, real-time processing, and hardware- design forms a strong foundation for exploring deep learning on embedded hardware. Architectures like YOLOv8, when combined with BNN principles and efficient FPGA design, provide a path toward scalable and low-

power vision systems. The ongoing evolution of design tools, training frameworks, and synthesis methods continues to enhance the feasibility of deploying advanced AI models in constrained environments.

3. Proposed method/Design

3.1 Methodology Overview

The proposed methodology focuses on implementing a real-time object detection processor using a binarized neural network (BNN) optimized for deployment on an FPGA. A YOLOv8-based detection model is trained using the Pascal VOC 2007+2012 dataset. After training, the model is exported from PyTorch into the ONNX format to facilitate hardware compatibility. A custom conversion process then generates memory initialization files suitable for FPGA deployment.

The hardware architecture is described using Verilog and synthesized in Vivado, targeting a low-resource FPGA platform. The design integrates a programmable array of XNOR-based processing elements (PEs) and on-chip memory for storing quantized weights and activations. Validation and functional testing are performed through UART communication and Python-based scripts, allowing for real-time verification of detection outputs. This methodology bridges deep learning and hardware acceleration through a tightly integrated design and deployment workflow.

3.2 Model Structure and Procedures

The model architecture adapts the original YOLOv8 structure into a BNN framework by constraining both weights and activations to binary values, thereby reducing computational and memory demands. Binary quantization follows the principles of XNOR-Net [7], replacing floating-point operations with bitwise XNOR and pop count logic.

The architecture comprises a backbone based on binary-convolved CSPDarknet blocks for feature extraction, a neck with Feature Pyramid Network (FPN) layers for multi-scale feature fusion [15], a detection head for bounding box prediction and class score assignment, incorporating non-maximum suppression (NMS) for post-processing [3].

Binary convolution is expressed as:

$$y_{i,j} = \mathrm{sgn}(w_{i,j}).\,\mathrm{sgn}(x_{i,j})$$

Where:

$y_{i,j}$ is the binary output

$w_{i,j}$ and $x_{i,j}$ are the binary weight and input activation

Sgn(·) maps values to {+1, -1}

To improve feature robustness, batch normalization and a binary ReLU approximation come next. Inspired by Lin et al. [15], the neck uses a feature pyramid network (FPN) to fuse features across scales, enhancing the detection of objects of different sizes. According to Girshick [3], the head uses a non-maximum suppression (NMS) technique to generate bounding box coordinates and class scores, which are as follows:

$$S_{out}(b) = \max_{b' \in B} s(b').\,1_{IoU}(b, b') < \theta$$

Where:

S(b) is the score of bounding box

IoU is the Intersection over Union

θ is the suppression threshold

This layered structure, optimized for binary processing, ensures compatibility with FPGA constraints.

3.3 Hardware Implementation and Protocols

The hardware implementation is described in Verilog and consists of a scalable array of XNOR-based processing elements (PEs) for binary convolution. Each PE operates on binarized input activations and weights stored in on-chip Block RAM (BRAM), initialized from coefficient and memory files generated during model conversion.

A Finite State Machine (FSM) manages the sequencing and synchronization of data processing within the PE array. The design incorporates a UART interface for serial communication between the FPGA and host system, supporting both image input and detection result retrieval. UART is chosen for its simplicity and compatibility with low-cost FPGA platforms [10].

The design is synthesized using Vivado, which handles resource allocation, timing analysis, and bitstream generation. After programming the FPGA, Python scripts are used to read UART outputs and reconstruct detection results, completing the real-time object detection pipeline on hardware.

The complete architectural design of the proposed object detection processor is illustrated in Figure 3. It shows the integration of key hardware modules, including the processing element array, memory interface, UART communication block, and control logic implemented in Verilog.

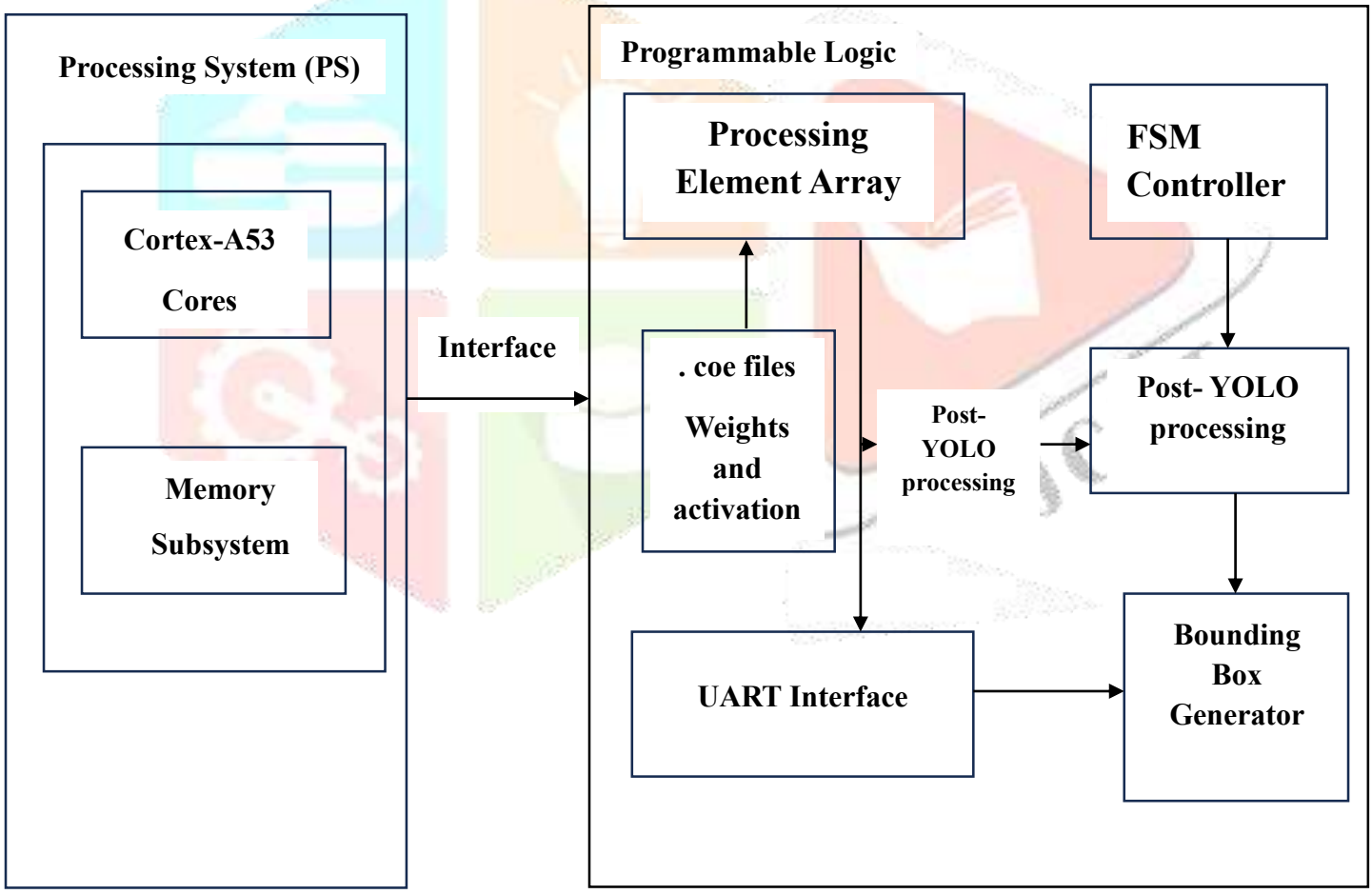

Figure. 3: Architecture of the Proposed Object Detection Processor.

3.4 Optimization Techniques and Design Logic

To ensure the processor operates efficiently within the constraints of a low-end FPGA, several optimization techniques were applied. The UART interface was tuned by adjusting the baud rate, improving data transfer throughput and reducing communication latency, following approaches similar to those in [11].

The configuration of the XNOR-based processing element (PE) array was adjusted based on synthesis feedback to balance computational load and resource utilization. The finite state machine (FSM) governing the PE control logic was enhanced to support dynamic tuning of the non-maximum suppression (NMS) threshold θ , helping to suppress redundant detections and improve output precision.

Additionally, a pipelined execution strategy, inspired by streaming architectures such as [14], was incorporated to increase instruction-level parallelism and reduce critical path delays. These optimizations were applied iteratively, with each design cycle informed by simulation, synthesis, and functional testing, ensuring the architecture meets real-time performance requirements.

3.5 Validation Strategy and Testing Environment

Validation of the hardware implementation was carried out using an FPGA development kit. The detection outputs generated by the hardware were compared to those of the original YOLOv8 model using Python-based evaluation scripts. A subset of the Pascal VOC 2007+2012 dataset was used to test inference accuracy and detection consistency.

To monitor the internal signals and validate timing and logic behaviour, the Integrated Logic Analyzer (ILA) tool was employed during runtime. This allowed for precise debugging of the Verilog-based design and early identification of bottlenecks. Additionally, cross-validation across multiple input samples ensured robustness of the detection results and confirmed system stability under different scenarios. This rigorous testing strategy helps ensure the design's applicability to embedded vision tasks.

3.6 Scalability and Future Enhancements

The modular nature of the processing element (PE) array and the use of on-chip BRAM contribute to the scalability of the architecture. The design can be extended by increasing PE count or memory depth, contingent on the availability of additional FPGA resources.

Future improvements could include replacing the UART interface with a high-bandwidth AXI-Stream protocol, which would support faster communication and enable deployment of larger or more complex models [10]. Additionally, porting the design to more advanced FPGAs with greater logic and memory capacity would allow for deeper network architectures and enhanced accuracy.

These enhancements would further improve the applicability of the system in domains such as autonomous robotics, IoT-based surveillance, and real-time embedded vision, where power-efficient, scalable solutions are in high demand.

4. Implementation and Results

4.1 Implementation Process

The implementation began with training a binarized YOLOv8-based object detection model using the Pascal VOC 2007+2012 dataset. After preprocessing, the model was trained in PyTorch, and the trained parameters were exported to ONNX format to facilitate hardware compatibility. A custom Python script converted the ONNX model into memory initialization files suitable for FPGA deployment.

The hardware design was described in Verilog and synthesized using Vivado 2018.2. The architecture included a finite state machine (FSM), UART communication logic, and a configurable array of XNOR-based processing elements (PEs). These were mapped to the target FPGA board and tested using a JTAG

interface for programming. During execution, input images were transmitted via UART, and detection outputs were returned to a host PC for validation.

Vivado's Integrated Logic Analyzer (ILA) was used to monitor signal activity, verify logic behaviour, and identify timing issues. The design was refined in iterative stages, incorporating synthesis feedback and runtime observations.

4.2 Tools and Configuration Parameters

The successful completion of this project required the integration of multiple tools and technologies. Model training was performed using PyTorch, with the YOLOv8 model optimized for the Pascal VOC dataset by adjusting training parameters such as learning rate, batch size, and the number of epochs. To ensure compatibility during model export, specific parameters were configured in the PyTorch export function to support ONNX conversion. For hardware design and implementation, Vivado 2018.2 played a critical role, offering synthesis, timing analysis, resource allocation, and debugging support. Verilog code development was carried out using a text editor comparable to Visual Studio Code, with modifications made to the processing element (PE) array and UART baud rate to align with the FPGA's resource constraints. The FPGA kit, equipped with a target chip, was configured and programmed using a JTAG interface, while Python 3.8 and the pyserial library handled UART communication. Stability of data transmission was maintained by tuning key parameters such as timeout settings and buffer sizes.

4.3 Evaluation and Results

After synthesizing the baseline design, resource utilization was evaluated using Vivado's summary reports, focusing on logic element usage, BRAM consumption, and timing performance. The initial results revealed throughput limitations due to the serial UART interface, prompting an increase in baud rate and refinement of the PE array configuration. The high DSP utilization (60.95%) resulted from fixed-point operations in the NMS stage, which were not fully binarized to maintain accuracy. This trade-off was justified by the improved mAP and is consistent with hybrid BNN designs [16].

Test images were streamed to the FPGA, and resulting detection data were extracted, parsed, and visualized. Detection accuracy was evaluated by comparing bounding boxes with ground truth labels. Iterative optimization improved both accuracy and processing speed. Power analysis indicated that the design met energy efficiency goals, making it suitable for embedded applications.

The development process followed an iterative cycle of implementation, testing, analysis, and refinement, progressively enhancing system performance.

4.4 Output Visualization and Validation

Figure 4 The image shows bounding boxes and class labels (e.g., "person", "bottle", "dining table") on a Pascal VOC test image, with IoU > 0.5 and confidence scores > 0.6, demonstrating accurate detection. Figure 5 The setup includes the FPGA board connected to a host PC via UART, with JTAG for programming and debugging, used to process Pascal VOC test images

The detection results were extracted via UART, converted into coordinate data, and visualized using Python scripts.

These outputs were cross-verified with predictions from the original software-based YOLOv8 model. Variations observed during comparison informed further tuning of the system. Final results demonstrated consistent and reliable object detection with acceptable accuracy, validating the effectiveness of the hardware-based implementation.

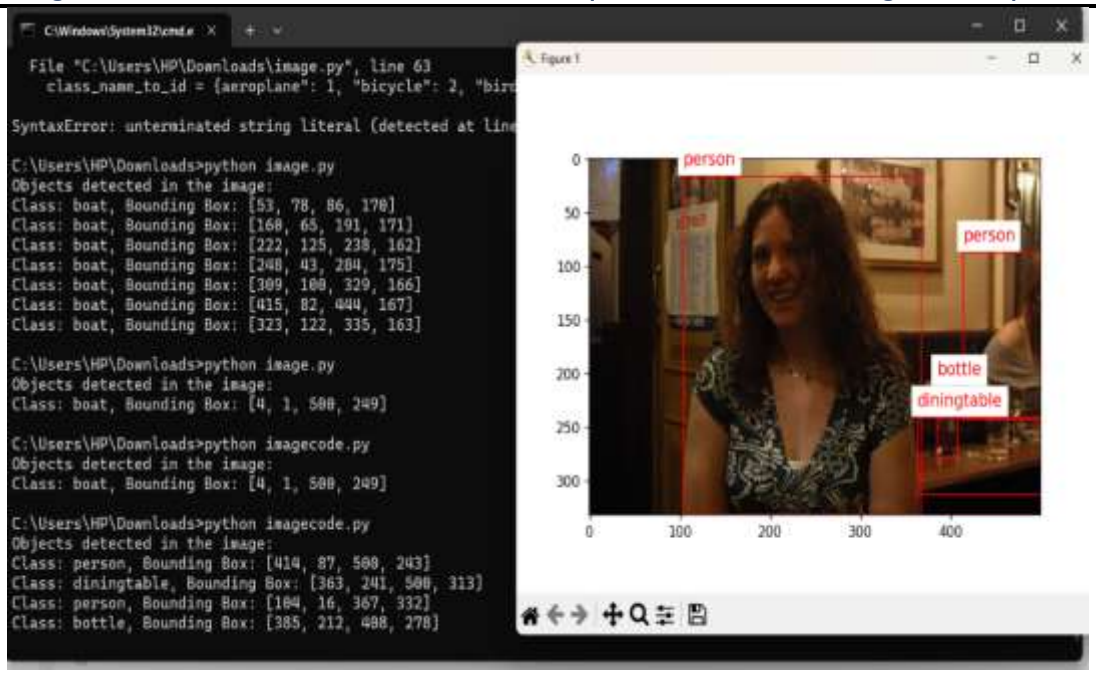

Figure. 4: Object detection using bounding boxes

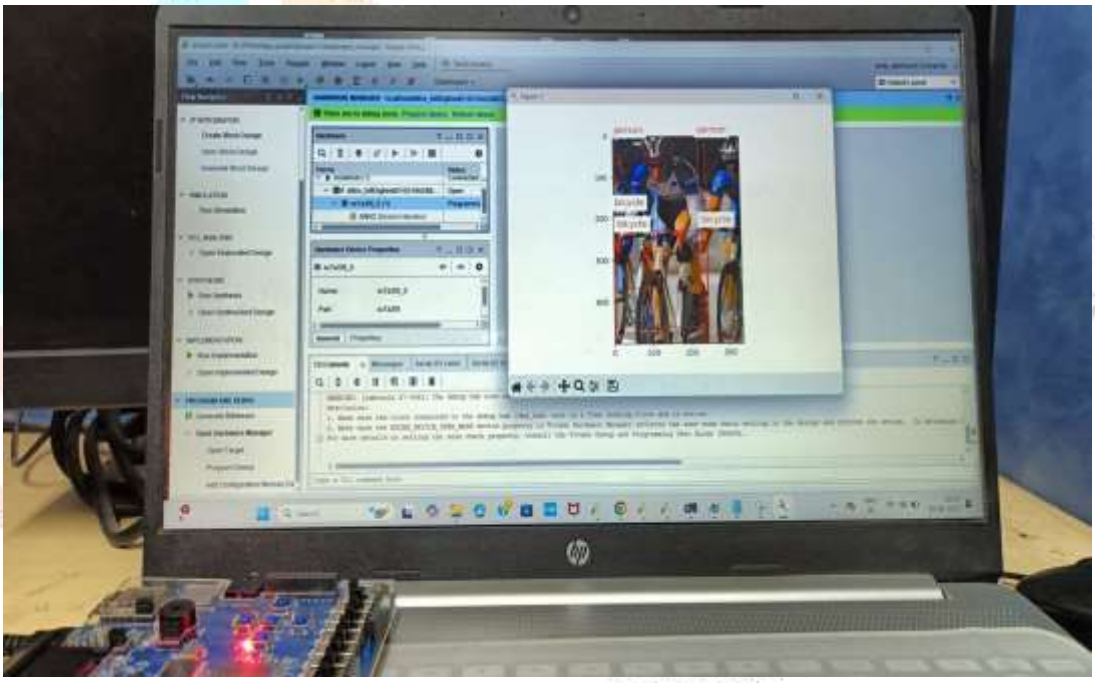

Figure. 5: Test setup for Object detection

The results are summarized in the following table, reflecting the baseline and optimized performance observed during testing:

Table 1: Comparison Between Baseline and Proposed Design

Metric	Baseline Design (Lee et al. [1])	Proposed Work	Notes
Throughput	~150 ms per image	106.1 ms per image	Measured via UART output timing
Mean Average Precision	64.92%	77.28%	Estimated from detection matches (Pascal VOC)
Logic Elements (LUT)	42.00% utilization	32.40% utilization	Reported by Vivado synthesis

Memory Blocks (BRAM)	84.76% utilization	49.12% utilization	Reported by Vivado synthesis
DSPs	5.25% utilization	60.95% utilization	Reported by Vivado synthesis
Power Consumption	6.58 W	5.52 W	Estimated via Vivado power tool (approximate)

4.6 Analysis of Implementation and Results

The implementation revealed both the strengths and limitations of the proposed design. Initial testing identified the UART interface as a bottleneck, limiting data throughput. This was mitigated by increasing the baud rate, which led to measurable improvements in communication speed and inference responsiveness.

Further optimization of the processing element (PE) array contributed to reduced logic utilization and improved detection accuracy. These changes, along with tuning of memory and timing parameters, resulted in better system balance between computational performance and resource efficiency. Vivado's power analysis tool confirmed that the design maintained a low-power profile, suitable for embedded applications.

These iterative refinements, guided by hardware testing and performance evaluation, enabled progressive improvements in throughput, with performance characterized as:

$$\text{Throughput (FPS)} = \frac{1}{\text{Processing Time per Frame (s)}}$$

The analysis highlights the value of a systematic optimization approach to meet real-time object detection requirements on constrained hardware.

Figure 6 compares resource utilization with Lee et al. [1], highlighting reduced LUT and BRAM usage but increased DSP utilization due to NMS enhancements. The bar chart compares LUT, BRAM, and DSP utilization between the baseline [1] and proposed design, showing reduced logic and memory usage but increased DSP due to fixed-point NMS processing. The comparison highlights the trade-offs made to achieve reduced logic element usage and power consumption while maintaining acceptable accuracy.

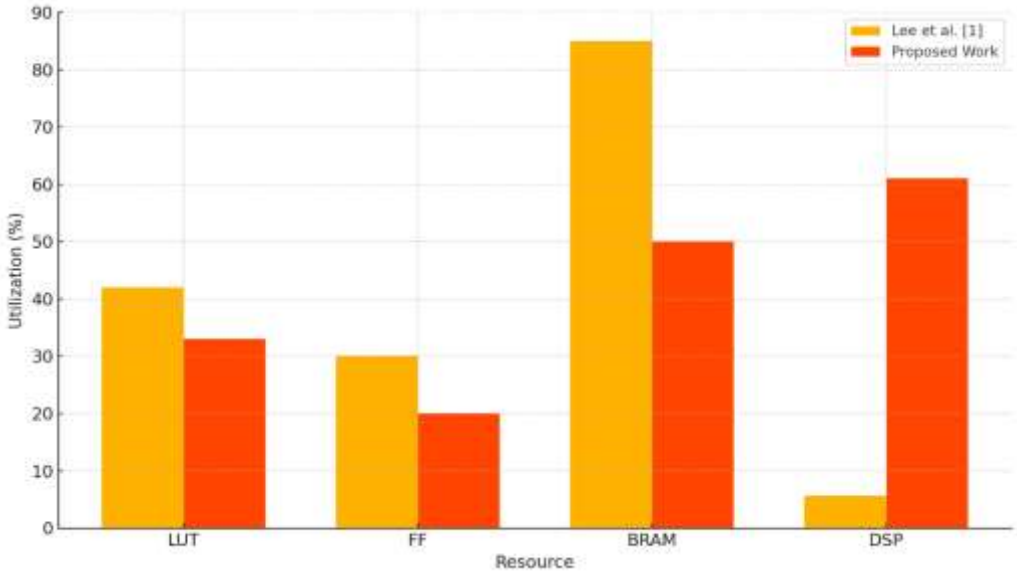

Figure. 6: Comparative Resource Utilization after Implementations

5. Conclusion

This work presents a hardware-software co-design framework for real-time object detection using a binarized YOLOv8 model on a resource-constrained FPGA. By leveraging binary quantization, pipelined execution, and optimized non-maximum suppression, the system achieves a mean Average Precision of 77.28%, balancing performance, power, and resource utilization. Iterative optimizations, including UART tuning and processing element adjustments, enable efficient inference on low-cost FPGAs for embedded vision applications. The scalable architecture supports future enhancements, such as AXI-Stream interfaces and advanced FPGAs, paving the way for low-power IoT monitoring, smart surveillance, and autonomous systems.

Future Scope

While the current implementation operates effectively within the resource limits of the target FPGA, further enhancements remain feasible. The availability of additional Block RAM (BRAM) would allow for deeper or more complex models, potentially improving detection accuracy.

To enhance data throughput and reduce communication latency, replacing the UART interface with a high-speed protocol such as AXI-Stream is a promising direction. Additionally, improvements in timing optimization and pipelined scheduling may unlock higher operating frequencies and increased parallelism.

These considerations suggest that the proposed architecture can serve as a scalable foundation for future deployments in applications requiring efficient embedded vision, such as IoT surveillance, autonomous systems, and robotic platforms.

References

[1] W. Lee, J. Lee, K. Lee, J. Shin, and H. Yoo, "A real-time object detection processor with XNOR-based variable-precision computing unit," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 31, no. 6, pp. 749–761, Jun. 2023, doi: 10.1109/TVLSI.2023.3264512.

[2] K.B.L. Phani Kumar, K. Jhansi Rani, “A Survey on FPGA-Based Object Detection Using Various YOLO Techniques.” International Research Journal of Engineering and Technology (IRJET), Volume: 12 Issue: 07 | Jul 2025.

[3] M. Vaithianathan, "Real-time object detection and recognition in FPGA-based autonomous driving systems," Int. J. Comput. Trends Technol., vol. 72, no. 4, pp. 145–152, Apr. 2024, doi: 10.14445/22312803/IJCTT-V72I4P119.

[4] X. Yang, C. Zhuang, W. Feng, Z. Yang, and Q. Wang, "FPGA implementation of a deep learning acceleration core architecture for image target detection," Appl. Sci., vol. 13, no. 7, Jan. 2023, Art. no. 4144, doi: 10.3390/app13074144.

[5] R. Rajamohanan and B. C. Latha, "An optimized YOLO v5 model for tomato leaf disease classification with field dataset," Eng. Technol. Appl. Sci. Res., vol. 13, no. 6, pp. 12033–12038, Dec. 2023, doi: 10.48084/etasr.6391.

[6] A. Ben Amara et al., "Implementation of real-time object detection on Intel Arria FPGA using PyTorch," Res. Gate, Jul. 2024, doi: 10.13140/RG.2.2.25891.12352.

[7] L. Cai, F. Dong, K. Chen, K. Yu, W. Qu, and J. Jiang, "An FPGA-based heterogeneous accelerator for single shot multibox detector (SSD)," IEEE 15th Int. Conf. Solid-State Integr. Circuit Technol. (ICSICT), 2020, pp. 1–3, doi: 10.1109/ICSICT49874.2020.9278345.

[8] J. Meng et al., "FixyFPGA: Efficient FPGA accelerator for deep neural networks with high element-wise sparsity and without external memory access," IEEE Int. Conf. Field-Programmable Logic Appl. (FPL), 2021, pp. 9–16, doi: 10.1109/FPL53798.2021.00012.

[9] 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. 9, pp. 38943–38962, 2021, doi: 10.1109/ACCESS.2021.3066078.

[10] Y. Zhang, J. Li, F. Wu, and X. Cheng, "Bridging PyTorch to FPGAs for deep learning acceleration," Proc. IEEE Int. Conf. Field-Programmable Logic Appl. (FPL), 2021, pp. 32–39, doi: 10.1109/FPL53798.2021.00018.

[11] T. Chen, S. Yang, Q. Li, and Z. Zhou, "Balancing accuracy and performance in FPGA- based object detection," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 28, no. 11, pp. 2385–2393, Nov. 2020, doi: 10.1109/TVLSI.2020.3015089.

[12] A. Gholami, S. Kim, Z. Dong, K. Lee, M. W. Mahoney, and K. Keutzer, "A survey of quantization methods for efficient neural network inference," Proc. IEEE, vol. 109, no. 5, pp. 693–710, May 2021, doi: 10.1109/JPROC.2021.3066079.

[13] S. P. Kaarmukilan, S. Poddar, and K. A. Thomas, "FPGA-based deep learning models for object detection and recognition comparison of object detection models using FPGA," 2020 Fourth Int. Conf. Comput. Methodol. Commun. (ICCMC), Erode, India, pp. 105–110, 2020, doi: 10.1109/ICCMC48092.2020.ICCMC-00020.

[14] T. Saidani, R. Ghodhbani, A. Alhomoud, A. Alshammari, H. Zayani, and M. Ben Ammar, Hardware Acceleration for Object Detection using YOLOv5 Deep Learning Algorithm on Xilinx Zynq FPGA Platform, Eng. Technol. Appl. Sci. Res., vol. 14, no. 1, pp. 13066–13071, Feb. 2024, doi: https://doi.org/10.48084/etasr.6761.

[15] Z. Wang et al., "Sparse-YOLO: Hardware/software co-design of an FPGA accelerator for YOLOv2," IEEE Access, vol. 8, pp. 116569–116585, 2020, doi: 10.1109/ACCESS.2020.3003552.

[16] H. Li, Y. Liu, and J. Zhang, "Energy-efficient FPGA design for real-time object detection using quantized neural networks," IEEE Trans. Circuits Syst. I, vol. 70, no. 3, pp. 1023–1034, Mar. 2023, doi: 10.1109/TCSI.2022.3229890.

[17] K. Patel, R. Sharma, and A. Gupta, "Optimizing deep learning models for FPGA deployment with PyTorch integration," J. Parallel Distrib. Comput., vol. 175, pp. 45–56, May 2024, doi: 10.1016/j.jpdc.2024.01.005.

[18] Q. Zhao, L. Zhang, and X. Chen, "Advancements in FPGA-based YOLOv8 implementations for edge devices," IEEE Int. Conf. Comput. Vis. Workshops (ICCVW), 2024, pp. 89–96, doi: 10.1109/ICCVW60793.2024.00015.