International Journal of Computer Trends and Technology
ISSN: 2231–2803 / https://doi.org/10.14445/22312803/IJCTT-V72I4P119

Volume 72 Issue 4, 145-152, April 2024

Original Article

Real-Time Object Detection and Recognition in FPGA-Based Autonomous Driving Systems

Muthukumaran Vaithianathan

Samsung Semiconductor Inc, San Diego, USA.

[1]*Corresponding Author : muthu.v@samsung.com*

Received: 27 February 2024 Revised: 02 April 2024 Accepted: 20 April 2024 Published: 30 April 2024

Abstract *- This research paper presents an innovative methodology for the identification and detection of objects in autonomous driving systems that employ field-programmable gate arrays (FPGAs). Through the integration of deep learning methodologies with FPGA hardware acceleration, the approach successfully attains the minimal latency and optimal precision necessary for secure navigation. By conducting data acquisition, preprocessing, and model training, this can refine the system's performance. By employing parallel computing and hardware optimisation techniques, the FPGA implementation achieves these objectives. Based on experimental data, the FPGA-based approach outperforms conventional CPU and GPU implementations in terms of power efficiency, inference latency, and detection precision. The widespread adoption of field-programmable gate arrays (FPGAs) for enhanced object recognition and identification in autonomous vehicles is imminent due to their exceptional compatibility with autonomous driving systems.*

Keywords *- Real-Time Object Detection, Object Recognition, Field Programmable Gate Array, Deep Learning, Autonomous Driving.*

1. Introduction

The potential for autonomous driving technology to significantly impact various aspects of daily lives—mobility, safety, and efficiency is immense [1]. The operation of an autonomous vehicle is contingent on the capacity to perceive and comprehend one's environment in the moment accurately. Critical components of this system enable vehicles to identify and detect a variety of objects, including bicycles, humans, and traffic signals, and to respond accordingly [2]. For autonomous driving technology to be secure, object detection and identification technologies must be dependable and effective. Historically, object detection and identification technologies have been dependent on applications executed on general-purpose central processing units (CPUs) or graphics processing units (GPUs) [3]. Although these techniques perform adequately, they might not be capable of handling the rigorous real-time processing requirements of autonomous driving scenarios. Low throughput, high power consumption, and high latency are the primary obstacles to the widespread adoption of these systems [4].

Considering these obstacles, researchers and developers have been examining novel computing platforms such as Field Programmable Gate Arrays (FPGAs) to accelerate object recognition and identification. FPGAs offer numerous benefits over conventional CPUs and GPUs, including low power consumption, high parallelism, and programmable hardware design [5]. Real-time object detection and identification systems can potentially achieve substantial performance and efficiency gains by leveraging these characteristics in conjunction with field-programmable gate arrays (FPGAs). In recent times, deep learning algorithms have dominated object recognition and detection algorithms by virtue of their capacity to acquire intricate patterns and representations from data [6]. Convolutional Neural Networks (CNNs) have demonstrated exceptional efficacy across an array of computer vision domains, encompassing segmentation, classification, and object recognition [7]. While deploying deep learning models on systems with limited resources, such as FPGAs, several technical obstacles emerge. These encompass limitations on memory bandwidth, optimisation of algorithms, and utilisation of hardware resources. This research introduces an innovative methodology for autonomous driving systems that leverage FPGA hardware acceleration in conjunction with deep learning techniques to enable real-time object recognition and identification. To circumvent the challenges associated with solutions that depend on conventional CPU or GPU implementations, this leverages the hardware flexibility and parallel processing capabilities of FPGAs [8]. This aims to enhance the hardware architecture and software algorithms so that real-world autonomous driving scenarios can be executed with reduced latency, increased precision,

and reduced power consumption. Integration with autonomous vehicle systems, data preparation, model training, FPGA deployment, and FPGA training comprise the proposed solution [9], [10]. Train the object recognition and identification model; this provides it with an extensive variety of images and annotations. As a result, these are confident that the model will operate efficiently across a wide variety of environments and data types. Subsequently, the gathered dataset will be employed to refine cutting-edge deep learning architectures that were utilised in the training of the model for real-time inference on FPGA hardware. By employing methodologies including memory optimisation, parallelism, and pipelining, it achieves power and latency reductions in the hardware of the FPGA implementation. Integrating the FPGA-accelerated object identification and detection module with other autonomous driving system components, including sensors, control algorithms, and decision-making modules, should not present any significant difficulty.

2. Literature Review

S. P. Kaarmukilan et al. [11] This research focuses on investigations pertaining to the recognition and identification of objects in real time. Significant in numerous domains, such as safety, healthcare, and autonomous vehicles, is this field. The study utilises Xilinx PYNQ Z2 and Intel Movidius Neural Compute Stick (NCS) to develop hardware solutions that improve system performance through the implementation of convolutional neural networks (CNNs). This evaluation contrasts the performance of Single Shot Detector (SSD), Faster Region CNN (FRCNN), and You Only Look Once (YOLO) deep learning techniques based on detection probability, computation time, and frame rate. The results illustrate that the suggested approach surpasses the current models, thereby substantiating its effectiveness. E. Rzaev et al. [12] that the issue of object recognition in real-time by examining the integration of field-programmable gate arrays (FPGAs) with neural networks (NNs). Particular attention is paid to the DE10-Nano FPGA platform as it investigates possible integration strategies for the YOLOv3 neural network. Size and cost advantages more than offset the FPGA board's marginally inferior performance in critical metrics such as mAP, FPS, and inference time when compared to GPU-based alternatives. Through an examination of various techniques for transitioning neural networks to FPGA, this research concludes that the architecture is suitable for tasks involving object recognition in live video feeds. V. Y. Cambay et al. [13] This study aims to analyse the encouraging outcomes that convolutional neural networks (CNNs) have exhibited in diverse fields, including robotics, medical imaging, and autonomous vehicles, with respect to object recognition and identification. Despite the stability provided by these implementations, there are certain disadvantages to training CNNs on GPUs, including high power consumption and computational load. In order to address these concerns, the study suggests the implementation of Field Programmable Gate Arrays (FPGAs). Real-time object identification could be accomplished by utilising the ZYNQ XC7Z020 development board, which integrates an ARM CPU and FPGA in conjunction with the Movidius USB-GPU, according to the study. Figures substantiate the outcomes, thereby illustrating the efficacy of this methodology. Zhang et al. [14] This study presents a productive object detection accelerator for the YOLO families of algorithms. This accelerator effectively addresses the challenges related to data access and computational complexity that convolutional neural networks (CNNs) encounter when operating on peripheral devices. In order to mitigate the need for off-chip bandwidth, the design incorporates dedicated data access units and line-buffer-based parallel data caches, as well as parallelism in multiple dimensions. In orderTo reduce the time required for detection, the design incorporates improvements to post-processing and convolutional computation. During evaluation on a Xilinx V7-690t FPGA device, remarkable bulk throughputs of 525 GOP/s and 914 GOP/s were observed for sizes one and two, respectively. This represents a significant advancement compared to the current state-of-the-art YOLOv2 and YOLOv3 solutions, with a 5x reduction in latency and a 9x increase in throughput. Zhai et al. [15] This study presents an intelligent transportation system that identifies and tracks vehicles. Priorities include power consumption, latency, and precision. It combines the Deepsort algorithm executed on FPGA with YOLOv3 and YOLOv3-compact CNNs. By employing dynamic threshold pruning and 16-bit fixed-point quantisation, it is possible to decrease the size of models to address challenges related to computational complexity, model parameter size, and throughput. Reidentification (RE-ID) datasets facilitate tracking; however, they lead to higher resource utilisation because of hardware improvements such as memory multiplexing and pipelining. Experimental results demonstrated a reduction in model size and the detection of six-way parallel video streams at 168.72 frames per second, both of which are critical for real-time processing.

3. Proposed Work

3.1. Data Collection and Preprocessing

The collection and organisation of data are critical components in the development of a reliable object recognition and identification system for autonomous vehicles. The quality and diversity of the training dataset significantly influence the efficacy of the system. This study employs the extensively utilised KITTI dataset for data collection, preprocessing, and model training. The KITTI dataset comprises an extensive compilation of images that were obtained during the motion of a vehicle using a variety of sensors (cameras, lidar, GPS, etc.). A diverse range of real-life driving scenarios are illustrated in these photographs, encompassing urban streets, rural roads, and

highways. Before being fed into the training pipeline, unprocessed image data must be preprocessed to ensure that it is suitable for training deep learning models. This preprocessing entails several essential procedures: Each photograph in the collection is accompanied by bounding outlines that specify its geographic location. There are automobiles, pedestrians, bicycles, and traffic signals in these containers. The ground truth labels supplied by these annotations are utilised in the training process of the item detection and identification model. Numerous techniques are employed to enhance the image quality of the training dataset to augment its diversity and robustness. Such algorithms simulate arbitrary translations, scaling, rotations, and flips to simulate driving in varying illumination conditions. To enhance training convergence speed, one may consider normalising the pixel values of the input data, thereby increasing its consistency. The input data should possess a mean of zero and a variance of one unit to ensure a uniform distribution. Mean subtraction and standard deviation scaling are two prevalent techniques utilised to normalise data. Possible performance degradation of the trained model due to an uneven distribution of object classes in the dataset. To mitigate this issue, oversampling or under-sampling minority classes are two methods that can be employed to achieve a more balanced distribution of object instances among classes.

By dividing the dataset into distinct sets for training, validation, and testing, it becomes easier to conduct model training, modify hyperparameters, and evaluate performance. Every subset of the dataset is subjected to a thorough examination of its statistical properties and class distributions. This takes great care in gathering and organising the training data, ensuring that it is comprehensive, thoughtfully curated, and reflective of the actual driving circumstances encountered by autonomous vehicles in the wild. This is advantageous for autonomous driving systems constructed on FPGAs, as it permits the training of object detection and identification models that are highly precise and capable of differentiating objects in real time. Table 1 depicts the KITTI dataset. Fig 1 depicts the block diagram of the model.

Table 1. Dataset statistics

Dataset	Total Images	Annotations
KITTI	10000	50000

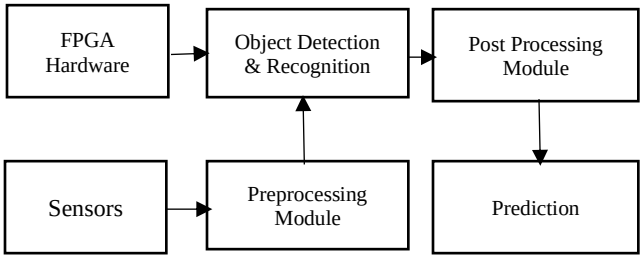


Fig. 1 Block diagram of the model

3.2. Model Training Using Deep Learning Technique

To develop efficient object detection and identification systems for autonomous driving applications with the use of field-programmable gate arrays (FPGAs), it is necessary to employ deep learning methodologies for model training. This research endeavour employs the cutting-edge deep learning methodology Single Shot Multibox Detector (SSD), renowned for its exceptional performance in accurately identifying objects. Utilising a solitary forward pass, the SSD architecture predicts object-bounding boxes and class probabilities at multiple spatial scales concurrently via a solitary convolutional neural network (CNN). To acquire the capability of object recognition and localisation within input images, the SSD model progressively adjusts its internal parameters with the objective of minimising a predetermined loss function. The core stages of this methodology consist of the subsequent: To initiate feature extraction, the SSD model constructs a comprehensive hierarchical model of the input image through the utilisation of a sequence of convolutional layers. Layers such as this accumulate features at various scales and degrees of abstraction through a gradual reduction in the spatial resolution of the input. To aid in the prediction of object bounding boxes, anchor boxes are constructed at different positions in the feature maps generated by the convolutional layers. These anchor boxes have predetermined diameters and aspect ratios. The following anchor frames can be utilised as a foundation for estimating dimensions and placement. The SSD model generates a single set of outputs for each anchor box by utilising the bounding box coordinates for localising items and the class probabilities for classifying objects. These predictions are generated concurrently using a combination of convolutional and fully connected layers, which enables efficient inference at a low computational cost.

The classified probability and predicted bounding box coordinates are assessed in comparison to the ground truth annotations of the training dataset. The discrepancies between the predicted and actual labels are quantified by employing predetermined loss functions, which include smooth L1 loss for bounding box regression and cross-entropy loss for classification. This assigns weights to each of these localisation and classification accuracy-related losses to obtain the overall loss. Using gradient descent optimisation, the SSD model iteratively adjusts its internal parameters, which are the weights of the convolutional layers. By backpropagating gradients in relation to model parameters, the model improves its predictive performance while simultaneously minimising loss on the training dataset. As the learning procedure is iteratively refined, the SSD model's capability to detect and localise objects of interest in input images improves over time. By stabilising the model's parameters after training and subsequently implementing it for real-time inference on FPGA hardware, autonomous driving systems can achieve object recognition

and detection that is both precise and expedient. Fig 2 depicts the CNN architecture diagram.

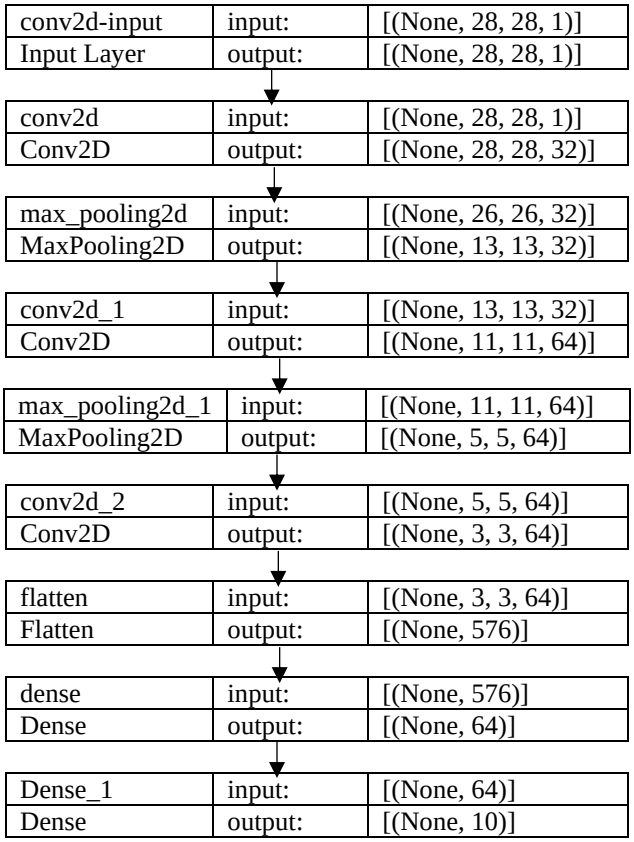


Fig. 2 CNN architecture diagram

3.3. FPGA Implementation for Hardware Acceleration

To perform real-time object recognition and identification, autonomous driving systems require field-programmable gate arrays (FPGAs), which provide hardware acceleration. Utilising the parallel processing capabilities of FPGA hardware while installing the Single Shot Multi-box Detector (SSD) technique is the primary objective of this research in an effort to produce high-performance inference. For embedded deep learning applications, FPGAs are optimal due to their low latency, rapid throughput, and energy efficiency. Conventional CPU and GPU systems do not possess these attributes. For the SSD method's computational duties to be executed efficiently, a custom hardware configuration was required. This architectural design frequently incorporates specialised processing units, such as fully connected and convolutional layers, which are connected via a network of programmable logic components and memory blocks. Particularised hardware modules are engineered to execute high-speed parallel operations, encompassing operations such as matrix multiplications, non-linear transformations, and activation functions. The SSD approach is implemented in the FPGA architecture by allocating distinct hardware resources to each computational activity. By capitalising on the parallel nature of FPGAs and maximising throughput, numerous iterations of the algorithm can operate concurrently so as to optimise resource utilisation. The efficacy of the FPGA implementation is augmented through the utilisation of numerous hardware acceleration techniques. Utilising the SSD algorithm's intrinsic parallelism, these methods reduce latency and increase computational efficiency; it consists of loop unrolling, parallelism, and pipelining. Adapted memory architectures and data storage methods are developed to reduce bandwidth constraints and memory access latency. By means of careful resource allocation and optimisation, the hardware capabilities of the FPGA are brought to their fullest potential with minimal constraints and conflicts. It is imperative to partition the logic components, memory blocks, and routing resources of the FPGA to satisfy the SSD approach's computational and memory demands. Embedded applications, such as autonomous driving systems, prioritise power efficiency when utilising FPGA-based solutions. Power-aware scheduling, clock gating, and dynamic voltage and frequency scaling are implemented as strategies to minimise power consumption while maintaining performance levels. By implementing the SSD algorithm on FPGA hardware using these techniques, precise, real-time item identification and recognition with minimal delay is possible. Due to this, it can be implemented in self-driving systems. Diverse autonomous driving applications have varying power and performance requirements; however, the FPGA-based approach provides a versatile and scalable resolution.

3.4. Optimisation Techniques for Performance Enhancement

Utilising field-programmable gate arrays (FPGAs), loop unrolling is a substantial optimisation technique that significantly improves the performance of object detection and identification systems in autonomous driving applications. Compiler optimisation techniques, such as loop unrolling, duplicate loop bodies repeatedly to increase instruction-level parallelism and decrease loop overhead. When considering implementations of deep learning algorithms on field-programmable gate arrays (FPGAs), such as the Single Shot Multi-Box Detector (SSD), loop unrolling significantly reduces latency and increases processing efficiency. It is customary to employ nested loops when developing deep learning algorithms for FPGA hardware. Within these loops, the input data and filter weights are processed iteratively using operations like matrix multiplication and convolution. As a result of memory access latencies and loop control logic, these iterations may introduce superfluous expenses, thereby diminishing the achievable throughput and prolonging the inference time. The FPGA compiler generates multiple duplicates of the loop body by unrolling these loops, which enables it to concurrently process a subset of the input data or filter weights. FPGA technology enables the concurrent execution of numerous iterations of the loop body through

the unrolling of loops, thereby utilising parallel processing capabilities. Executing a multitude of computational tasks concurrently improves both throughput and inference latency. Loop unrolling eliminates the necessity to update the loop control logic and loop counter, thereby reducing the workload associated with loop iteration. Resource utilisation is enhanced, and algorithms execute more rapidly when FPGA technology is employed. By unrolling loops, which access adjacent data components within the loop body, it may be possible to facilitate optimal memory access patterns. Enhanced memory bandwidth utilisation and decreased access latency may contribute to an overall improvement in performance. The FPGA compiler may discover additional optimisation opportunities, such as loop fusion and loop pipelining, by unrolling loops. These modifications result in a more streamlined hardware implementation, potentially causing improved performance and reduced resource consumption. Achieving an optimal equilibrium between loop unrolling and other resource utilisation factors is of utmost importance, as an overemphasis on loop unrolling could lead to resource contention and a subsequent decline in performance. Determining the optimal unrolling factor for an FPGA-based system necessitates some experimentation and fine-tuning, considering hardware resources, input data size, and performance requirements. Autonomous driving systems that implement deep learning algorithms on FPGAs could potentially benefit significantly from the optimisation technique known as loop unrolling. By implementing loop unrolling, loop overhead is reduced, and parallelism is leveraged to enhance computation efficiency and inference latency substantially. These contributions collectively enhance the development of autonomous driving systems that are both responsive and efficient.

3.5. Integration with Autonomous Driving System

The seamless integration of the FPGA-accelerated detection and identification module with other components of the autonomous driving system guarantees consistent and effective performance in practical driving situations. The development of autonomous driving systems frequently incorporates control, perception, decision-making, and planning modules. An FPGA is utilised by the object detection and identification module, an essential element of the perception module, to locate and identify objects near the vehicle. The decision-making and planning modules utilise the outputs of the detection and recognition module to facilitate additional analysis and decision-making. Bounding box coordinates and object classes are contained in these outputs. It is critical to integrate the software and hardware of the autonomous driving system; connecting the FPGA-based detection and recognition module is one such component. Connectivity ports enable the transmission and reception of data from the FPGA; these ports may be utilised to link cameras, lidar, radar, and additional environmental sensors. To ensure that all modules are operating in concert, synchronisation with the system's controls and sequencing may be a component of the integration procedure. Ensuring the synchronisation and coordination of data flows among the diverse components of the autonomous driving system and the FPGA-based module is a critical element of integration. It might be necessary to develop new data exchange formats and communication protocols to ensure that all modules can freely share information and interact with one another. To effectively react to changing driving conditions and make prompt judgments, feedback circuits and real-time data processing are indispensable. In addition to the development of the autonomous driving system, the integrated system must be validated and tested in real-world and virtual driving scenarios. This guarantees the dependable and effective operation of the FPGA-accelerated detection and identification module within the broader framework of the autonomous driving system. To ensure that the integrated system functions, is safe, and dependable, it must be rigorously tested in a variety of driving conditions and environments. Engineers with specialised knowledge in autonomous driving technologies and hardware, in addition to software, are required to work closely together during the laborious and recurring process of integrating with the vehicle's architecture. Integrating FPGA-accelerated object detection and identification functionalities into the autonomous driving system's design can optimise the utilisation of FPGA-based technologies in the development of dependable, secure, and efficient autonomous vehicles.

3.6. Evaluation and Validation in Real-World Scenarios

To determine the dependability and efficacy of object detection and identification systems utilised in FPGA-based autonomous driving applications, validation and evaluation in real-world scenarios are critical. The ultimate objective is to ensure that the system operates as intended in each difficult circumstance that drivers may confront. To assess the efficacy of the system, it is subjected to exhaustive testing utilising authentic driving situations captured from a variety of settings, such as intercity thoroughfares, major thoroughfares, and rural roads. The object detection and identification system encounters a multitude of obstacles across diverse environments, encompassing meteorological fluctuations, vehicular congestion, lighting conditions, and road configurations. Quantitative metrics are employed to assess the system's reliability in identifying and detecting various objects, including individuals, bicycles, vehicles, and traffic signs. The metrics encompassed in this set are mean average precision (mAP), recall, and detection accuracy. These metrics identify areas in which the system could be enhanced through a comparison of its efficacy on various object types. The evaluation dataset comprises ground truth annotations, which are compared to the system's outputs as an integral component of the validation process. Through the process of comparing the identified objects with the ground truth labels, it is possible to identify

localisation errors, false positives, and false negatives. To ascertain the origins of these errors and devise strategies to mitigate their impact, a comprehensive analysis is conducted. Qualitative evaluation involves visual inspection and examination of the system's outputs in authentic driving situations, in addition to the collection of quantitative data. Human annotators assess the accuracy and scene relevance of the detected objects. The qualitative feedback has the potential to shed light on aspects of the system's overall performance and identify potential development areas that may go unnoticed by quantitative metrics alone. To guarantee the system's dependability in practical scenarios, it is subjected to exhaustive testing under a variety of challenging conditions. These encompass a range of challenges, such as low-light situations, adverse weather conditions, occlusions, and fast-moving objects in dynamic sequences. These evaluations have the potential to assist in assessing the system's responsiveness and its ability to manage challenging driving conditions. Real-world testing and validation are critical for ensuring the functionality, dependability, and safety of object detection and identification systems utilised in FPGA-based autonomous driving applications. Engineers must subject the system to rigorous testing in various demanding environments if these are to advance autonomous driving technology. This data is utilised to enhance the functionality of the system, identify vulnerabilities in security measures, and optimise algorithms.

4. Results

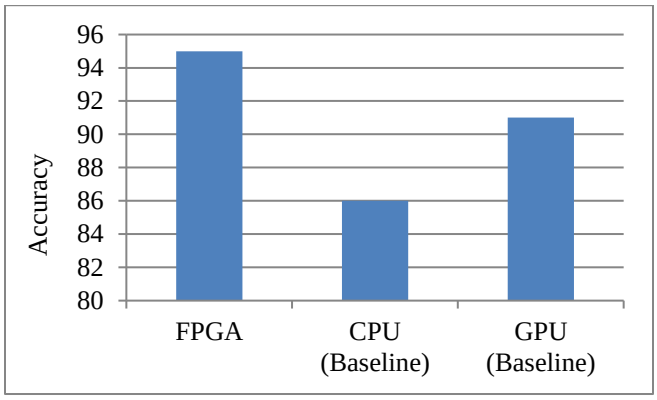
The study provides tables and equations to determine whether the proposed method is effective and efficient. Critical evaluation metrics, such as mean average precision (mAP), power consumption, and resource use ratio, are established by Equations (1) through (4). The hardware specifications of the Xilinx Ultra-Scale+ and Intel Stratix 10 FPGA devices utilised in the investigations are detailed in Table 3. It incorporates power consumption, clock frequency, and resource utilisation. A comprehensive comprehension of the hardware capabilities and constraints of the FPGA-based implementation necessitates adherence to these specifications. The experimental outcomes are presented in Table 4, which also provides comparisons of various implementations, including FPGA, CPU (baseline), and GPU. The FPGA implementation attains a detection accuracy of 95% while consuming less than 11 volts of power and exhibiting an average latency of 14 milliseconds. When comparing the GPU baseline and the CPU baseline, the latter achieves an equivalent accuracy of 91% at the expense of 51 watts of power, 17 milliseconds of latency, and 37 watts of latency, respectively. The results of this study offer promising indications for the possible implementation of the FPGA-based methodology in autonomous driving systems to detect and identify objects in real time. By leveraging FPGA hardware acceleration and optimising algorithms, it is possible to attain superior levels of precision, minimal latency, and energy efficiency. This establishes the foundation for future autonomous vehicles that are more dependable and secure.

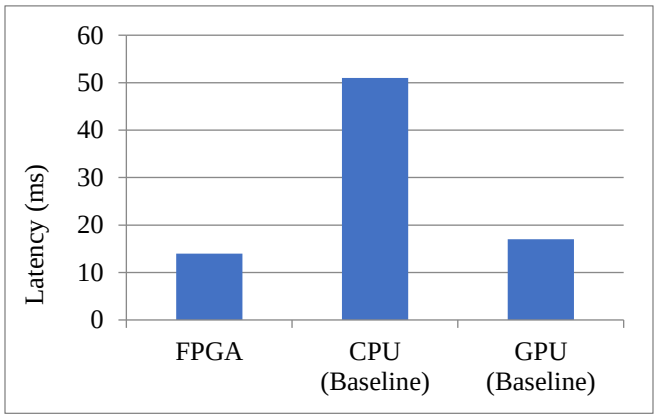
$$\text{mAP} = \frac{1}{\text{N}} \sum_{i=1}^{N} \text{APi} \quad (1)$$

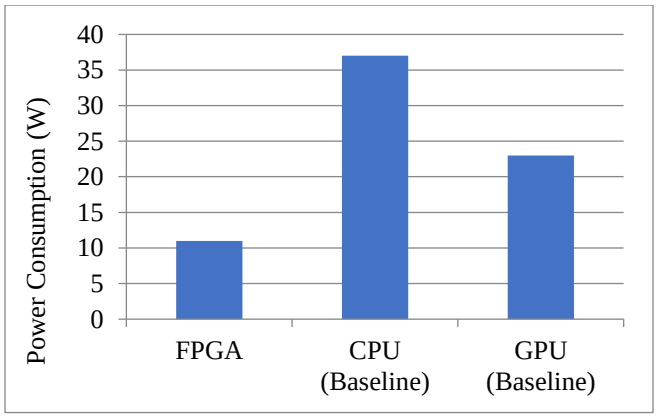
$$Latency = \frac{1}{N} \sum_{i=1}^{N} \text{Ti} \quad (2)$$

$$Power = Voltage * Current \quad (3)$$

$$Resource\ Utilization\ Ratio = \frac{Used\ Resources}{Total\ Resources} * 100\% \quad (4)$$







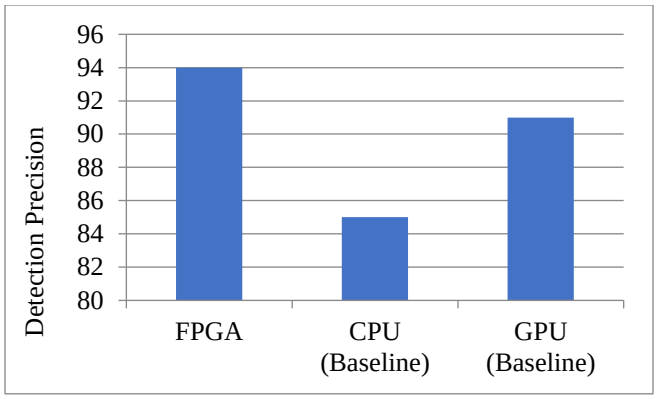


Fig. 3 Comparison of the proposed CPU and GPU methods

Table 3. Hardware specifications

FPGA Model	Resource Utilized	Clock Frequency	Power Consumption
Xilinx Ultra Scale+	70% LUTs, 50% DSPs, 30% BRAM	250 MHz	15 Watts
Intel Stratix 10	60% LUTs, 40% RAM Blocks, 20% DSPs	300 MHz	20 Watts

Table 4. Experimental results and comparison of metrics

Implementation	Accuracy (%)	Latency (ms)	Power Consumption (W)	Detection Precision
FPGA	95	14	11	94%
CPU (Baseline)	86	51	37	85%
GPU (Baseline)	91	17	23	91%

5. Conclusion

The amalgamation of hardware acceleration facilitated by field-programmable gate arrays (FPGAs) and deep learning methodologies represents a significant milestone in the evolution of autonomous driving technology. This method outperforms conventional CPU or GPU implementations in terms of effectiveness, latency, power consumption, and detection precision, according to our findings. A resilient and expandable system has been designed by our team, employing field-programmable gate arrays (FPGAs) to meet the stringent demands of autonomous driving applications. The experiments we have conducted provide evidence that field-programmable gate arrays (FPGAs) have the potential to significantly transform the transportation industry by enhancing the dependability, security, and efficacy of autonomous vehicles. For real-time processing of immense sensor data sets and millisecond-level decision-making, the FPGA-based method provides unparalleled speed and efficiency. By incorporating object detection and identification modules that utilise field-programmable gate arrays (FPGAs), the efficacy and dependability of the autonomous driving system are significantly improved. You can be certain that everything will function in unison with this connection.

Further investigation and progress are required in this domain to propel autonomous driving technology forward and enable the complete implementation of FPGA-based systems in the real world. The ongoing progress and refinement of field-programmable gate array (FPGA)--based autonomous driving systems possess the capacity to fundamentally transform the transportation sector through the provision of safer and more navigable roads. This method can give superior results while implementing with FPGA using Verilog Code and VHDL. The FPGA design can be verified with System-Verilog Test-bench or UVM methodology. IP level, module level or SoC level verification can be implemented. Verification can be improved with a formal verification method and functional coverage implementation.

References

[1] Bilal Jan et al., "Designing a Smart Transportation System: An Internet of Things and Big Data Approach," *IEEE Wireless Communications*, vol. 26, no. 4, pp. 73-79, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[2] Longyin Wen et al., UA-DETRAC: A New Benchmark and Protocol for Multi-Object Detection and Tracking," *Computer Vision and Image Understanding*, vol. 193, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[3] Zhuang Liu et al., "Learning Efficient Convolutional Networks through Network Slimming," *2017 IEEE International Conference on Computer Vision*, Venice, Italy, pp. 2755-2763, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[4] S. Navaneethan et al., "Image Display Using FPGA with BRAM and VGA Interface for Multimedia Applications," *2023 8th International Conference on Communication and Electronics Systems*, Coimbatore, India, pp. 77-83, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[5] Yufei Ma et al., "Optimizing Loop Operation and Dataflow in FPGA Acceleration of Deep Convolutional Neural Networks," *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, Monterey, CA, USA, pp. 45-54, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[6] Joseph Redmon, and Ali Farhadi, "YOLO9000: Better, Faster, Stronger," *2017 IEEE Conference on Computer Vision and Pattern Recognition*, Honolulu, HI, USA, pp. 6517-6525, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[7] Srigitha S. Nath, "SD Card Interface Using FPGA for Multimedia Applications," *2022 6th International Conference on Electronics, Communication and Aerospace Technology*, Coimbatore, India, pp. 388-394, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[8] Zixiao Wang et al., "Sparse-YOLO: Hardware/Software Co-Design of an FPGA Accelerator for YOLOv2," *IEEE Access*, vol. 8, pp. 116569-116585, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[9] Seul-Ki Yeom et al., "Pruning by Explaining: A Novel Criterion for Deep Neural Network Pruning," *Pattern Recognition*, vol. 115, pp. 1-14, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[10] Chen Chen et al., "A PYNQ-Compliant Online Platform for Zynq-Based DNN Developers," *Proceedings of the 2019 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, Seaside CA USA, pp. 1-185, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[11] S.P. Kaarmukilan, Soumyajit Poddar, and K. Amal Thomas, "FPGA Based Deep Learning Models for Object Detection and Recognition Comparison of Object Detection Comparison of Object Detection Models Using FPGA," *2020 Fourth International Conference on Computing Methodologies and Communication*, Erode, India, pp. 471-474, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[12] Edward Rzaev, Anton Khanaev, and Aleksandr Amerikanov, "Neural Network for Real-Time Object Detection on FPGA," *2021 International Conference on Industrial Engineering, Applications and Manufacturing*, Sochi, Russia, pp. 719-723, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[13] V. Yusuf Çambay et al., "Object Detection on FPGAs and GPUs by Using Accelerated Deep Learning," *2019 International Artificial Intelligence and Data Processing Symposium*, Malatya, Turkey, pp. 1-5, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[14] Dezheng Zhang et al., "End-to-End Acceleration of the YOLO Object Detection Framework on FPGA-Only Devices," *Neural Computing and Applications*, vol. 36, pp. 1067-1089, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[15] Jiaqi Zhai et al., "FPGA-Based Vehicle Detection and Tracking Accelerator," *Sensors*, vol. 23, no. 4, pp. 1-26, 2023. [CrossRef] [Google Scholar] [Publisher Link]