INTERNATIONAL JOURNAL OF SCIENTIFIC RESEARCH IN ENGINEERING AND MANAGEMENT (IJSREM)
VOLUME: 10 ISSUE: 01 | JAN - 2026 SJIF RATING: 8.586 ISSN: 2582-3930

FPGA Based Hardware Accelerated Object Tracking

Amith M
Electronics & Communication Engineering
Jss Academy of Technical Education
Affilated to VTU, Belagavi
Bengaluru, Karnataka, India
amitmmmaruthi@gmail.com

Bhuvan V Prasad
Electronics & Communication Engineering
Jss Academy of Technical Education
Affilated to VTU, Belagavi
Bengaluru, Karnataka, India
bhuvanvprasad@gmail.com

Chandrakantha
Electronics & Communication Engineering
Jss Academy of Technical Education
Affilated to VTU, Belagavi
Bengaluru, Karnataka, India
chandrakanthasn867@gmail.com

Gagan P Acharya
Electronics & Communication Engineering
Jss Academy of Technical Education
Affilated to VTU, Belagavi
Bengaluru, Karnataka, India
gaganpacharya1158@gmail.com

Project Guide
Dr. Anuradha M G
Associate Professor
Jss Academy of Technical Education
Affilated to VTU, Belagavi
Bengaluru, Karnataka, India
anuradhamg@jssateb.ac.in

***Abstract*—Object tracking is used in various day to day applications such as face recognition devices, CCTV camera applications, vehicle tracking systems. This paper presents the implementation of Tiny-Yolo V3 algorithm on the PYNQ-Z2 FPGA board for achieving hardware accelerated detection model for images and video inputs. Here we make use of the both software and hardware components and integrate them for optimal results. Initially the model was implemented and tested on CPU only model, then integrated hardware-assisted control modules on the FPGA fabric to support real-time operation. The proposed work achieves effective object detection with low power consumption. Future works will include DPU based full-YOLO network deployment.**

***Keywords*—FPGA, Tiny-Yolo V3, Object tracking, PYNQ-Z2, Hardware Acceleration, DPU**

I. INTRODUCTION

Image detection is one of the most essential processes in computer vision, where the goal is to identify and classify objects present within a static image. It forms the foundation for many advanced applications such as medical diagnostics, industrial automation, autonomous vehicles, and security monitoring. Conventional image detection systems that rely on central processing units (CPUs) or graphics processing units (GPUs) often struggle to achieve real-time performance due to their high computational demands and power consumption. As a result, these systems are not always suitable for embedded or low-power environments.

To address these limitations, Field Programmable Gate Arrays (FPGAs) have gained significant attention as a hardware acceleration platform for image detection. FPGAs feature a reconfigurable logic structure that supports parallel data processing, allowing multiple operations to be executed simultaneously. This architectural advantage enables faster computation, reduced latency, and lower energy consumption compared to traditional processors.

Implementing image detection algorithms such as YOLO (You Only Look Once) on FPGA-based platforms like the PYNQ-Z2 allows for efficient real-time image processing without compromising accuracy. The combination of hardware parallelism and algorithmic optimization makes FPGA-based image detection systems a promising solution for intelligent vision applications that demand both high performance and energy efficiency.

II. RELATED WORKS

A. A Reconfigurable CNN-Based Accelerator Design for Fast and Energy-Efficient Object Detection System on Mobile FPGA.

This article presents a highly reconfigurable FPGA hardware accelerator for CNNs, optimized for speed and power efficiency. Techniques like minimized data transfer, controlled pipeline design, and low-power RTL techniques (clock gating, OR-based MAC architecture) were applied. Implemented on a PYNQ-Z1 mobile FPGA-SoC, the accelerator showed 15% higher throughput, 16% lower power consumption, and 58% better hardware utilization compared to baseline. It achieved 9.17 FPS for object detection, demonstrating feasibility for real-time processing on mobile FPGAs.

B. *Adaptive Subsampling for ROI-Based Visual Tracking: Algorithms and FPGA Implementation*

This work explores adaptive subsampling in image sensors for energy-efficient embedded vision systems. The approach combines object detection (using YOLO or ECO tracker) with a Kalman filter for ROI prediction. Implemented on an FPGA, the ECO tracker-based algorithm achieves competitive accuracy and power efficiency (4W power consumption, 19.23 FPS) compared to YOLO-based approach (6W power consumption).

C. *An FPGA Accelerator for High-Speed Moving Objects Detection and Tracking With a Spike Camera.*

This paper proposes a neural-inspired scheme for ultra-high-speed object detection and tracking using a spike camera. A parallelized filtering module and divided detection module are designed to accelerate the algorithm, with hardware optimizations to reduce operations and resource consumption. Implemented on a Xilinx ZCU-102 board, the accelerator achieves 19x speedup and processes over 20,000 spike images per second.

D. *Real-Time SSDLite Object Detection on FPGA.*

This article proposes a novel hardware architecture and system optimization techniques for real-time DNN-based object detection. A Neural Processing Unit (NPU) with heterogeneous units and a Task Control Unit (TCU) are designed to accelerate neural networks, reduce memory accesses, and increase utilization. Implemented on Intel FPGAs, the system achieves higher throughput, lower latency, and better energy efficiency while maintaining high detection accuracy, outperforming previous state-of-the-art works

E. *A Low-Cost High-Speed Object Tracking VLSI System Based on Unified Textural and Dynamic Compressive Features.*

This paper presents a low-cost, high-speed object tracking VLSI system that utilizes unified textural and dynamic compressive features and elliptic matching. The system features a memory-centric architecture with multiple-level pipelines and parallel processing circuits, enabling efficient processing. Implemented on an FPGA, it achieves 600 frames/s for 320x240 resolution images at 100 MHz clock frequency, making it suitable for high-speed, low-cost embedded visual tracking applications.

III. BACKGROUND

In this section, we first explain the Tiny-Yolo V3 detection algorithm and then elaborate the other operations and other postprocess.

A. *Tiny-Yolo V3 Overview*

YOLOv3 (You Only Look Once, Version 3) is a real-time object detection algorithm that identifies specific objects in videos, live feeds, or images. The YOLO machine learning algorithm uses features learned by a Deep. Tiny-YOLOv3 is a stripped-down version of YOLOv3 with fewer layers and a smaller network size, making it significantly faster for real-time applications.

Convolutional Neural Network to detect objects located in an image. Joseph Redmon and Ali Farhadi created the first version of YOLO algorithms in 2016. The two later released Version 3 two years later, in 2018. YOLOv3 is an improved version of YOLO and YOLOv2. YOLO is implemented using the Keras or OpenCV deep learning libraries.

YOLO is a Convolutional Neural Network (CNN) , a type of deep neural network, for performing object detection in real-time. CNNs are classifier-based systems that process input images as structured arrays of data and recognize patterns between them. YOLO has the advantage of being much faster than other networks and still maintains accuracy.

It allows the object detection model to look at the whole image at test time. This means that the global context in the image informs the predictions. YOLO and other CNN algorithms “score” regions based on their similarities to predefined classes.

High-scoring regions are noted as positive detections of whatever class they most closely identify with. For example, in self-driving car footage, YOLO can be used to detect different kinds of vehicles depending on which regions of the video score highly in comparison to pre-defined classes of vehicles. This scoring mechanism, involving regional proposals, enables precise and efficient object detection across various scenes.

B. *Yolo V3 Architecture*

The YOLOv3 algorithm first separates an image into a grid. Each grid cell predicts some number of bounding boxes (sometimes referred to as anchor boxes) around objects that score highly with the aforementioned predefined classes.

Each bounding box has a respective confidence score of how accurate it assumes that prediction should be. Only one object is identified per bounding box. The bounding boxes are generated by clustering the dimensions of the ground truth boxes from the original dataset to find the most common shapes and sizes.

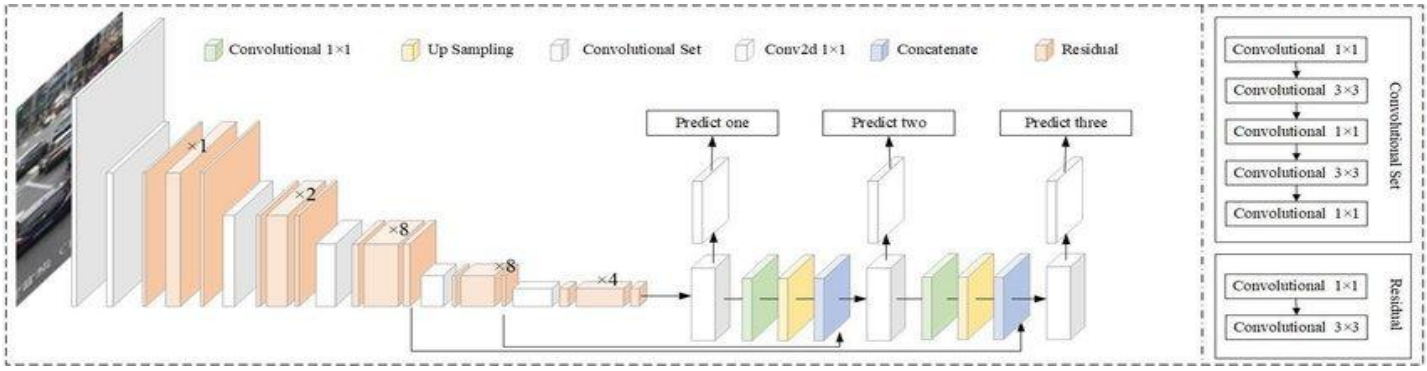


Fig 1. - How YOLO v3 works

Other comparable algorithms that can carry out the same objective are R-CNN (Region-based Convolutional Neural Networks made in 2015), Fast R-CNN (R-CNN improvement developed in 2017), and MASK R-CNN. However, unlike systems like R-CNN and Fast R-CNN, YOLO can perform classification and bounding box regression at the same time.

C. Field Programmable Gate Array (FPGA): Overview

Field Programmable Gate Array (FPGA) is an integrated circuit designed to be configured by the user after manufacturing. Unlike traditional processors such as CPUs and GPUs, which have fixed hardware structures, an FPGA offers reconfigurable hardware that allows developers to define and modify the internal circuit behavior according to specific application requirements. This flexibility makes FPGAs highly suitable for tasks that demand parallel processing, low latency, and energy-efficient computation.

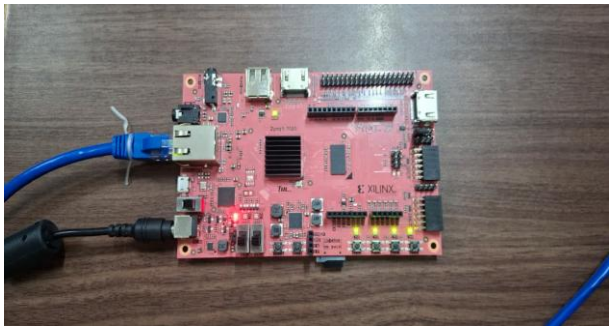


Fig 2.-FPGA Board circuit

D. How FPGA Works

The operation of an FPGA is based on configuring its internal logic blocks and interconnections to perform a desired digital function. When a design is implemented, it is first described using a Hardware Description Language (HDL) such as VHDL or Verilog. This HDL code defines how data should flow through the hardware. The design is then synthesized and programmed onto the FPGA using a configuration file known as a bitstream. Once programmed, the FPGA acts as a custom digital circuit capable of executing multiple operations in parallel. This allows FPGAs to deliver high-speed performance while maintaining flexibility to reprogram the device for new applications.

E. FPGA Architecture

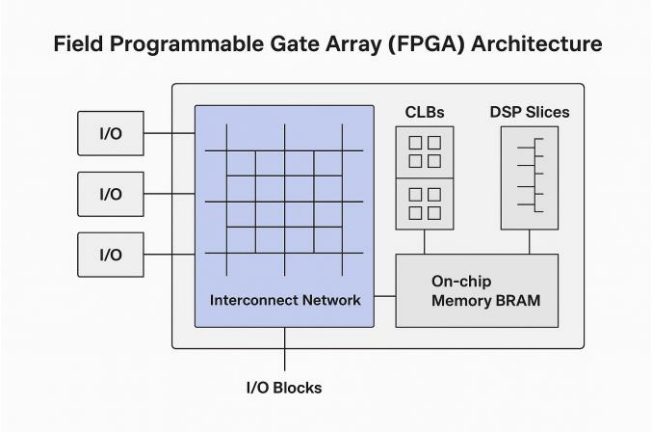


Fig 3. - FPGA Architecture

The internal structure of an FPGA consists of several key components:

1. Configurable logic Blocks (CLBs): These are the primary building units of an FPGA. Each CLB contains look-up tables (LUTs), flip-flops, and multiplexers that can be configured to implement logical and arithmetic operations.
2. Input/Output Blocks (IOBs): These blocks manage communication between the FPGA and external devices or peripherals. They control the direction,

voltage levels, and timing of input and output signals.

3. Interconnect Network: A flexible routing system that connects the various logic blocks, memory units, and I/O blocks. The interconnect allows for customizable data paths based on the user's design.
4. Digital Signal Processing (DSP) Slices: These dedicated blocks handle high-speed arithmetic operations such as multiplication and accumulation, making them ideal for signal and image processing applications.
5. On-chip Memory (BRAM): Block RAM modules provide fast and temporary data storage that supports parallel data access during computation.
6. Clock Management Units: These circuits generate and distribute clock signals to synchronize operations across the FPGA.

IV. METHODOLOGY

The system implements a hybrid hardware–software approach for real-time object detection and tracking on an FPGA-based platform. Video frames captured from a USB camera are processed by the processing system, where object detection and classification are performed using a lightweight YOLO-based model. A custom hardware IP developed using Vitis HLS is integrated into the programmable logic to assist control and improve system stability. The final output with bounding boxes and labels is displayed through other display interfaces.

A. USB Camera (Input)

The USB camera acts as the primary input source for the system and is responsible for capturing visual data from the surrounding environment. It continuously acquires video frames in real time and forwards them to the processing system. The camera operates at a fixed resolution suitable for real-time processing, ensuring a balance between image quality and computational requirements. The captured frames serve as the raw input for subsequent processing and analysis stages.

B. Processing System (Frame Processing)

The processing system receives the raw video frames from the USB camera and performs initial frame-level operations. These operations include frame acquisition, resizing, format conversion, and basic preprocessing required to prepare the frames for further analysis. This stage ensures that the input data is in a consistent and optimized format, enabling efficient interaction between software components and hardware-assisted modules.

C. Object Detection & Tracking

In this stage, the processed frames are analyzed to identify and track objects of interest across consecutive frames. The system determines the presence and movement of objects within the scene and maintains continuity of detected objects over time. This stage plays a crucial role in reducing redundant computations by ensuring that only relevant frames and regions are forwarded for detailed classification, thereby improving overall system efficiency.

D. Hardware-Assisted Control (FPGA)

The hardware-assisted control module, implemented on the FPGA, accelerates selected control and decision-making operations that are computationally intensive when executed purely in software. By offloading these operations to dedicated hardware logic, the system achieves improved processing speed and reduced load on the processing system. The FPGA operates in coordination with the processor through a control interface, enabling efficient hardware–software co-operation.

E. YOLO-Based Classification

The YOLO-based classification stage performs object recognition and labeling on selected frames. Using a lightweight object detection model, the system classifies detected objects into predefined categories such as persons and vehicles. Bounding boxes and class labels are generated based on the classification results. This stage provides semantic understanding of the scene while maintaining acceptable performance for near real-time operation.

F. Output Display

The final stage presents the detection and tracking results to the user. The output includes video frames annotated with bounding boxes, object labels, and relevant visual indicators. The processed output is displayed through a browser-based or software interface, allowing real-time observation of system performance. This stage enables validation of detection accuracy and overall system functionality.

G. Block diagram

The block diagram illustrates the overall architecture of the proposed real-time object detection and tracking system. Video input is captured using a USB camera and forwarded to the processing system for initial frame handling and preprocessing. The processed frames are then analyzed for object detection and tracking, after which selected, operations are accelerated using hardware-assisted control implemented on the FPGA. YOLO-based classification is applied to identify and label detected objects, and the final annotated output is displayed to the user. This structured flow enables efficient cooperation between software and hardware components, improving processing performance while maintaining detection accuracy.

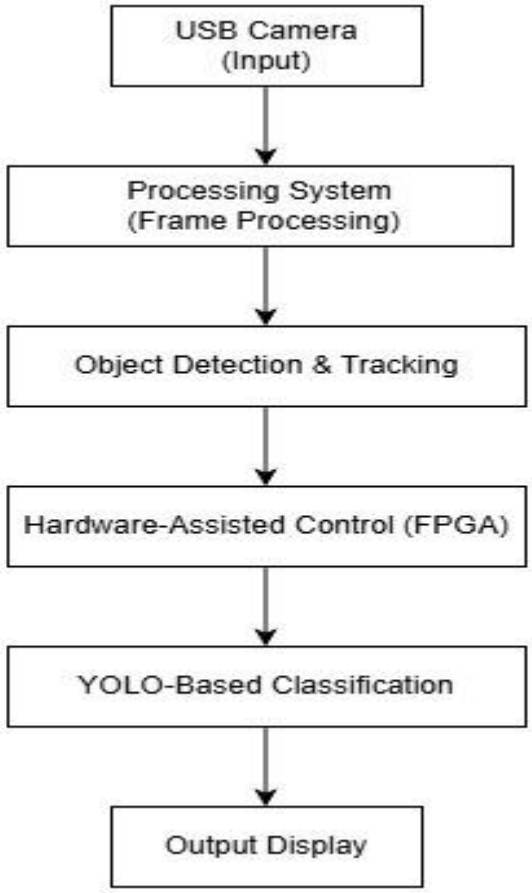
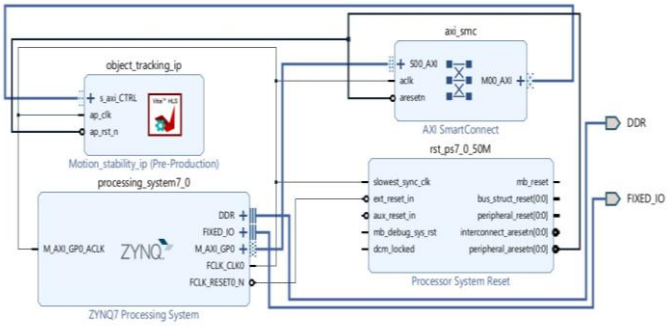


Fig 4. - Block diagram

H. Vivado Block Diagram



The hardware-assisted control module is developed using Vitis High-Level Synthesis (HLS) and integrated into the system using the Vivado Design Suite. Vitis HLS is used to describe the required control functionality using a high-level C/C++ based approach. The design is synthesized to generate a custom hardware intellectual property (IP) core.

The generated IP core is imported into Vivado, where a block design is created that includes the Processing System and the custom hardware IP. Necessary clock, reset, and control connections are established to ensure proper operation. After validation of the block design, a bitstream is generated and programmed onto the FPGA

I. Results

The proposed real-time object tracking system was implemented and evaluated on an FPGA-based platform using both image datasets and live video input captured through a USB camera. The system was tested under different scenarios to analyze its detection capability, tracking stability, and real-time performance. During testing, the system successfully detected and tracked multiple objects within a single frame. Common object classes such as persons, cars, bicycles, and two-wheelers were correctly identified and highlighted using bounding boxes along with corresponding class labels. The detection results obtained from static images demonstrate that the system can identify objects of varying sizes and orientations with reasonable accuracy

Fig.5.-Input image

Fig.6.-Output image

The CPU-based image detection experiments demonstrated accurate identification of objects such as persons and vehicles; however, the average processing time for a single image was approximately 10 seconds, indicating a low inference speed. In contrast, the real-time implementation

using the hybrid hardware–software approach achieved a stable frame rate in the range of 10–12 frames per second under live camera input. The system was observed to operate reliably without frame drops during continuous execution, demonstrating improved responsiveness compared to the CPU-only approach. These results highlight the effectiveness of hardware-assisted processing in supporting real-time object detection and tracking

V. CONCLUSION AND FUTURE WORK

The implementation of an FPGA-based hardware-accelerated object tracking system using the Tiny-YOLO algorithm demonstrates the feasibility of deploying deep learning models on low-power embedded platforms. The proposed design efficiently combines software control on the ARM processor with hardware acceleration on the FPGA fabric to achieve faster inference and reduced energy consumption compared to CPU-only systems. Experimental results verify that the developed framework provides reliable person and object detection performance suitable for intelligent surveillance and real-time vision applications.

Future work will focus on enhancing the system by implementing full model quantization and integrating a Deep Processing Unit (DPU) for complete YOLO inference on the FPGA. Further optimization using pruning and pipelined hardware architectures can improve throughput and resource utilization. Extending the design to support multi-object tracking, real-time video analytics, and cloud-edge interoperability will strengthen its applicability in advanced embedded AI and smart surveillance domains.

VII REFERENCES

[1]W. Shi, X. Li, Z. Yu, and G. Overett, “An FPGA-Based Hardware Accelerator for Traffic Sign Detection,” IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 25, no. 4, pp. 1362–1373, Apr. 2017. DOI: 10.1109/TVLSI.2016.2631428

[2] V. H. Kim and K. K. Choi, “A Reconfigurable CNN-Based Accelerator Design for Fast and Energy-Efficient Object Detection System on Mobile FPGA,” IEEE Access, vol. 11, pp. 59438–59452, Jun. 2023. DOI: 10.1109/ACCESS.2023.3285279

[3] L. Chang, S. Zhang, H. Du, Y. Chen, and S. Wang, “A Reconfigurable Neural Network Processor With Tile-Grained Multicore Pipeline for Object Detection on FPGA,” IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 29, no. 11, pp. 1967–1979, Nov. 2021.DOI: 10.1109/TVLSI.2021.3109580

[4] S. Kim, S. Na, B. Y. Kong, J. Choi, and I.-C. Park, “Real-Time SSDLite Object Detection on FPGA,” IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 29, no. 6, pp. 1192–1204, Jun. 2021.DOI: 10.1109/TVLSI.2021.3064639

[5] Q. Gu, T. Takaki, and I. Ishii, “Fast FPGA-Based Multiobject Feature Extraction,” IEEE Transactions on Circuits and Systems for Video Technology, vol. 23, no. 1, pp. 30–43, Jan. 2013.DOI: 10.1109/TCSVT.2012.2202195

[6] M. A. S. Kamal and S. K. Nandi, “A Low-Cost, High-Speed Object Tracking VLSI System Based on Unified Textural and Dynamic Compressive Features,” IEEE Transactions on Circuits and Systems for Video Technology, vol. 29, no. 10, pp. 2842–2854, Oct. 2019.

[7] Y. Chen et al., “Acceleration of Rotated Object Detection on FPGA,” IEEE Access, vol. 11, pp. 102948–102960, 2023.

[8] C. Zhang, P. Li, and Y. Wang, “Optimizing FPGA-Based Accelerator Design for Deep Convolutional Neural Networks,” Proceedings of the ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA), 2015, pp. 161–170.

[9] J. Redmon and A. Farhadi, “YOLOv3: An Incremental Improvement,” arXiv preprint arXiv:1804.02767, 2018.

[10] Xilinx, “PYNQ: Python Productivity for Zynq,” Xilinx Developer Documentation, 2023. [Online]. Available: https://pynq.io

[11] X. Zhang et al., “DPU-PYNQ: An End-to-End Deep Learning Accelerator on PYNQ-Z2 FPGA,” IEEE International Conference on Field Programmable Logic and

[12] A. Abdelouahab et al., “Accelerating Deep Convolutional Networks on FPGA with Fixed-Point Quantization,” IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 37, no. 11, pp. 2755–2768, 2018.

[13] M. Suda et al., “Throughput-Optimized FPGA Accelerator for YOLO Object Detection,” IEEE International Conference on Reconfigurable Computing and FPGAs (ReConFig), 2020.

[14] H. Nakahara and T. Yonekawa, “FPGA-based YOLO Real-time Object Detection System,” IEEE International Conference on Consumer Electronics (ICCE), 2019.

[15] K. Venieris and C. Bouganis, “f-CNN: An FPGA-based Framework for Convolutional Neural Networks,” IEEE International Conference on Field-Programmable Logic and Applications (FPL), 2016.

[16] T. Chen et al., “Eyeriss: An Energy-Efficient Reconfigurable Accelerator for Deep Convolutional Neural Networks,” IEEE Journal of Solid-State Circuits, vol. 52, no. 1, pp. 127–138, Jan. 2017.

[17] S. I. Venieris et al., “Toolflows for Mapping Convolutional Neural Networks on FPGAs: A Survey and Future Directions,” ACM Computing Surveys, vol. 51, no. 3, pp. 1–39, 2018.

[18] P. Ma et al., “High-Efficiency FPGA Accelerator for YOLO Object Detection,” Electronics, vol. 10, no. 2, pp. 145–158, 2021.

[19] Z. Bai et al., “Low-Power FPGA Accelerator for Tiny-YOLO Object Detection,” IEEE Access, vol. 9, pp. 57628–57639, 2021.

[20] W. Liu et al., "SSD: Single Shot MultiBox Detector," European Conference on Computer Vision (ECCV), 2016, pp. 21–37.