Journal of VLSI and Embedded System Design
Journal of VLSI and Embedded System Design • ISSN 3139-1818 • Jan - Apr • 2 • 1 • 2026 (pp. 21-33)
RESEARCH ARTICLE

Hardware–Software Co-Design Framework for High-Performance Embedded Vision Processing on Reconfigurable FPGA Architectures

N. Disages K. Mojail[1*], H.R. Mira[2], H. Taconi[3], D. Gravino[4], P.K. Nestaris[5]

[1-5]Centro de Investigacion y Desarrollo de TecnologiasAeronauticas (CITeA) FuerzaAerea Argentina Las Higueras, Cordoba,Argentina

KEYWORDS:
Embedded vision processing,
FPGA architecture,
hardware–software co-design,
reconfigurable computing,
embedded AI,
real-time image processing.

ARTICLE HISTORY:
Received : 17.01.2026
Revised : 15.02.2026
Accepted : 19.03.2026

Abstract

Embedded vision systems have currently become an intrinsic part of contemporary intelligent technologies as they offer real-time perception and decision-making processes within autonomous vehicles, smart surveillance, industrial automation, robotics, and Internet of Things (IoT) controlled devices. The application of these applications involves effective processing of massive amounts of visual data on built-in platforms, where very tight limits on computational resources, power consumption and memory capacity tend to restrict the performance of standard processing architectures. Owning to its real-time vision algorithms and especially those that rely on deep learning and other advanced image processing methods, such algorithms require high computing throughput and low latency, which cannot be easily attainable solely using general-purpose processors. This paper presents a hardware software co-architecture deployed to reconfigurable FPGA systems to overcome the problem of high-performance embedded vision processing. The suggested model produces the combination of algorithmic optimization and hardware acceleration to programme vision processing activities efficiently on the FPGA resources. The framework can get the computational workloads divided in both software control modules and hardware accelerators, thereby facilitating parallel processing, improvement of dataflow and lesser overhead of memory access. The architecture takes into account dedicated processing units of Image preprocessing, feature extraction and classification, streaming pipeline and optimised memory map hierarchy to facilitate continuous real time image processing. The offered co-design solution can enhance the system performance in a tremendous way due to a decrease in the processing latency, enhanced throughput and acceptable energy consumption in the embedded setup. Within the experimental analysis, it is shown that the FPGA-based architecture performs a significant enhancement in the real time processing power than traditional CPU based implementations, with a higher frame processing rate at reduced power. The outcome of the research shows that the suggested hardware-software co-design structure can provide a scalable and efficient design of the next-generation embedded vision systems with a high performance with severe resource limitations.

How to cite this article: NMojail NDK, Mir HR, Taconi H, Gravino D, Nestaris PK· Hardware–Software Co-Design Framework for High-Performance Embedded Vision Processing on Reconfigurable FPGA Architectures. Journal of VLSI and Embedded System Design. Vol. 2, No. 1, 2026 (pp. 21 -33).

Introduction

Embedded vision system is now a key part of the contemporary intelligent electronics systems that allows devices to sense and process visual data at the border of the computing environments. They combine all three visualisation acquisition, processing and decisions capabilities within the small embedded platforms, and enable real-time visual analysis with minimal dependence on cloud-based resources. Embedded vision is broadly used in many applications including robotics, autonomous driving, smart camera, industrial inspection, healthcare monitoring and the Internet of Things (IoT). Embedded vision is used in robotics and autonomous systems to detect objects, navigate, and perceive the environment, and smart sur-

veillance systems to monitor and provide security-related functions. The high rate of development of intelligent devices and cyber-physical systems has consequently placed greater attention on efficient embedded processing solutions of vision that can operate in large amounts of visual data within a reduced latency, and high reliability.[1, 11]

Even with such tremendous developments in computer vision algorithms, it is difficult to apply such methods in embedded systems. The latest vision algorithms, especially those involving deep learning and convolutional neural networks entail extensive computations that include convolution operations, feature analysis and multiplication of matrices and nonlinear functions of activation. Recurrent operations are very demanding with respect to computational throughput and memory bandwidth. Nonetheless, embedded systems are normally constrained in terms of their resources which involve low processing power, low memory and restrictive energy use. High computational loads preferably in the power or energy efficient embedded system might considerably waste power and thermal energy dissipation. Also, numerous embedded vision applications need to be real-time capable, and have the visual data analysed in the shortest possible time, to allow using it in time-intensive decisions. The traditional general-purpose processors suggest are commonly unsuited to fulfil such demands due to the fact that the sequential nature of instruction execution and the prevalent access operations frequency limit the performance of general-purpose processors in dealing with large vision workloads.[4, 5, 14]

As a solution to these problems, reconfigurable computing devices like Field-Programmable Gate Arrays (FPGAs) have become a major focus as efficient devices to use as Hardware accelerators when implementing embedded vision systems. FPGAs offer a very parallel and generalised computing design capable of permitting the designers to actualize special hardware circuit's specific to particular computational activities. In contrast with general-purpose processors, which execute each instruction sequentially FPGA architectures allow a number of processing functions to be undertaken simultaneously using specific datapaths and parallel pipelines. Such parallelism permits image processing tasks that are computationally intensive to be handled more effectively and overall decreases processing latency. In addition, FPGA devices are highly configurable and system designers can exploit hardware resources that include logic blocks, memory modules and digital signal processing units to fit the application. All these features render FPGAs quite appropriate to speed up convolution operations, feature extraction phases, and other recurrent computations that are common in embedded vision algorithms.[1, 9, 10]

Nevertheless, the efficient utilisation of the embedded vision systems based on FPGA processors cannot be obtained by merely porting algorithms to hardware resources. The common design approaches of the traditional design approach can be divided into the programming of software algorithms and hardware implementation, which can reduce the efficient use of computational resources. To remove this shortcoming, hardware-software co-design has sprung up as a successful method and approach to co-efficiently optimise the structures of algorithms and hardware architectures when building systems. In a co-design process, the computational workloads are divided between the program code that executes on encased processor units as well as the hardware devices that are introduced into the FPGA fabric. Definite changes in algorithms, like performance inaccuracy, data moving strategies, and parallel work designs can be added to enhance a fit to hardware applications. Meanwhile, hardware architecture can be implemented in such a way that it facilitates the effective transfer of data, access of memory and pipeline execution. The integrated design approach is much more effective in enhancing the efficiency of the system, lowering the energy consumption, and ensuring that in embedded computing systems, real-time performance can be determined on a strict basis.[3, 6]

Out of these opportunities and challenges, this study hypothesises a hardware-software co-design framework on high-performance embedded vision processing through reconfigurable FPGA architecture. The suggested system would concentrate on the combination of optimised vision analysis algorithms and custom designs of FPGA hardware modules to display an effective image processing in real time. The design presents a streamlined vision pipeline that enables the image preprocessing and feature development and classification processes with reduced computational cost. Moreover, a reconfigurable FPGA-based accelerator architecture is created to take advantage of parallel processing and execution of dataflow. An optimised memory hierarchy and pipeline-based system is also introduced in the proposed system to minimise latency and enhance throughput when performing the continuous image working process. Through experimental evaluation, it has been shown that the proposed hardware-software co-design system can provide great processing speed, energy efficiency, and real-time performance over traditional systems based on processors in terms of vision systems. The findings point to the capability of FPGA-based co-design methodologies in supporting scalable and energy-efficient embedded vision systems in intelligent systems of the next generation.[6, 15, 16]

Related Work

The mushrooming of embedded vision applications has stimulated the development of specialised modelling of hardware architectures that can engage visual information with reasonable resource limits. A number of studies have studied the implementation of FPGA, hardware acceleration of image processing and hardware-software co-design processes in order to enhance the performance of embedded vision systems. This part summarises the key works that occurred in these disciplines and what research gaps have inspired the given framework. FPGAs have enjoyed a thorough investigation as a convince tool in a potential platform of executing embedded vision algorithms because they are available to encourage parallel processing and tailorable hardware architecture. Initial studies have shown that an accelerator based on FPGA may significantly accelerate the time taken by the convolutional neural network and other vision algorithms by several folds with respect to general-purpose processors. Most notably, as an example, Chen et al. suggested the Eyeriss architecture, a low-energy convolutional-neural-networks accelerator, was able to show the power of special-purpose hardware to accelerate high-throughput visual computing workloads.[1] Conversely, that is the same case when Cong and Xiao proposed optimization methods to reduce the computational complexity of convolutional neural networks and allow them to be effectively implemented on hardware such as FPGAs.[2] These papers demonstrating the advantages of hardware acceleration in visual processing tasks were largely interested in CNN computation and not in entire embedded vision pipelines.

The use of FPGA based vision systems in real time vision systems like autonomous navigation, inspection systems in industries and surveillance systems has been on the increase over the last few years. Streaming pipelines with reconfigurable FPGA architectures provide the ability to implement image-processing pipelines to the designer: e.g. filtering, edge detection, feature extraction and object recognition. Research has shown that FPGA platforms can have high throughput with relatively low power consumption, and hence they can be used in the embedded vision applications deployed at the edge of the network. Recent studies also indicate that with the help of FPGA-based accelerators, deep learning models applied in vision tasks are also efficient. Indicatively, modern object detection models like YOLO have been suggested to be implemented with FPGA to support real-time industrial vision systems that achieve better performance and energy efficiency.[16] Also, surveys on event based vision processing platforms show how FPGA architectures may be utilised to facilitate asynchronous vision processing algorithms which greatly use less latency and less power in embedded systems.[10] The other field of study that is essential is the creation of hardware accelerators that are specifically designed as image processing and deep learning inference hardware devices. Convolutional neural networks are popularly used to perform visual recognition but they are computationally expensive thus making them difficult to implement in embedded systems. A number of authors have come up with specialised accelerators to enhance the efficiency of CNN computations. The scale of the neural network models have been minimised by using deep compression methods to improve storage and computing needs using pruning, quantization, and encoding methods.[5] Equally, less precise computation techniques have been investigated in order to reduce the arithmetic complexity and maintain a reasonable model accuracy.[4] The methods allow effective neural network behaviour mapping on hardware (FPGA-based accelerators). Besides, there are a number of accelerator architectures suggested to maximise the utilisation of data reuse and memory access pattern in convolution operations, the most expensive part of deep learning-based vision systems.[1, 9]

In addition to hardware acceleration, scholars have also paid more attention to methods of hardware-software co-design used to optimise embedded vision systems. Conventional system design methods tend to isolate algorithm design and hardware implementation and they may result in the inefficiency of resource utilisation in the implementation of complicated vision algorithms on embedded hardware. Hardware-software co-design can deal with this problem by co- optimizing software algorithms and hardware architectures when developing a system. Design methodologies like ESP4ML that use platforms offer frameworks of integrating embedded machine learning workloads into system-on-chip design platforms and keep them flexible and scalable [3]. Likewise, other more recent examples of successfully dividing workloads between software and hardware components have been shown to be very efficient with FPGA-based co-design methods that have been shown to substantially enhance system performance on edge inference inferences [6]. Automated mapping Software tools have also been suggested to encode high-level algorithms of image processing into optimised hardware platforms, minimising the design complexity and development time.[7] These changes show that co-design methods are becoming critical towards effective implementations of advanced vision algorithms over embedded platforms.

Even in case existing research has attained a good level in terms of FPGA-based vision systems and hardware accelerators, there are still numerous limitations. However, the most common focus of many of the past

studies lies on optimization of particular components of a vision processing pipeline, e.g. convolution operations, or a particular neural network architecture. Also, certain accelerator architectures are specialised to certain algorithms which makes them less flexible in the adaptation to other vision tasks or datasets. The second weakness is that many embedded vision systems are poorly integrated where their software control logic and hardware processing modules do not work together. In the absence of effective hardware-software co-design, data movement overhead, memory bottlenecks and pipeline inefficiencies may lead to decreased performance of the entire system. In addition, other FPGA designs focus on computational capabilities without considering any power savings and scalability needs in embedded and edge computing. In order to emphasise the properties of the representative FPGA-based embedded vision systems and hardware accelerators, Table 1 provides the overview of a number of existing solutions and compares their design goals and constraints.

As indicated in Table 1, the past studies have mainly been aimed at enhancing the computational efficiency of particular algorithm/application. Nonetheless, an overall system architecture which incorporates hardware/software co-design, and scaled FPGA implementation with the needed features to handle full embedded vision pipelines is still required. It is necessary to fill this gap to facilitate the realisation of flexible, high-performance vision processing in constrained embedded systems. The proposed work hence seeks to come up with a hardware-software co-design methodology that co-optimises the algorithm design, hardware architecture, and dataflow management of real-time embedded vision processing on reconfigurable FPGA platforms.

Embedded Vision Processing Pipeline

Embedded vision systems make use of structured processing pipeline whereby raw image data obtained by sensors are converted to useful information which can be utilised in making autonomous decisions. To achieve the benefits in real-time embedded system, this pipeline needs to be efficient and strict requirements concerning computational resources, power consumption and processing latency must be met. An average embedded vision pipeline comprises of various steps which are image acquisition, preprocessing, feature extraction, algorithmic inference and output generation. All the stages are significant because they are used to ready visual information to analyse it and make sure that the system can handle image streams without the need to halt. The general design of the in-house vision processing pipeline applied in this project is demonstrated in Figure 1 that displays the chain of processing steps that includes generating images, processing the images to generate a classification output.

Image acquisition and preprocessing is the first phase

Table 1: Comparison of Existing FPGA-Based Embedded Vision and Hardware Accelerator Approaches

Reference	Architecture Type	Application	Key Advantages	Limitations
Chen et al. [1]	CNN Hardware Accelerator	Deep neural network inference	Energy-efficient architecture	Focuses mainly on CNN computation
Cong & Xiao [2]	CNN Optimization Framework	Neural network acceleration	Reduced computation complexity	Limited system-level pipeline integration
Han et al. [5]	Model Compression Techniques	Deep learning inference	Reduced model size and computation	Not hardware-specific
Huang et al. [9]	Algorithm–Hardware Co-Design	CNN acceleration	Improved hardware compatibility	Limited flexibility for diverse vision tasks
Yan et al. [16]	FPGA-based YOLO Accelerator	Real-time object detection	High throughput for industrial vision	Architecture optimized for specific model
Kryjak [10]	FPGA Vision Survey	Event-based vision processing	Low latency asynchronous vision systems	Mostly focused on event cameras

Fig. 1: Embedded vision processing pipeline showing camera input, preprocessing, feature extraction, classification, and output stages.

of the pipeline and visual information is obtained by an image sensor like a CMOS imaging module. The sensor interface will convert the analogue signals on the camera into digital image frames which could be processed by the embedded system. Communication protocols normally implemented in this process include communication tools like MIPI, SPI or parallel camera interfaces. Once the images have been acquired, they are pre-processed to enhance the quality of data and minimise noises prior to further processing. The usual preprocessing methods are image normalisation, noise filtering, grayscale conversion, and resizing. The pre-processes are aimed at standardising input data and complexifying the calculation that will be made later in the pipeline.

The second step, after preprocessing, is feature extraction where the objective is to extract useful patterns or structures of the visual data. The extraction techniques convert raw pixel values into small relevant features, which bring out significant characteristics of the image. Classical feature extraction tools are edge detection, spatial filtering, determination of gradient and texture. Sobel or Canny philtres are edge detection algorithms that are often used to identify object boundaries in pictures. Moreover, philtre methods can also be used to sharpen certain image details through elimination of noise and highlighting of important details. Key image structures can also be represented using feature descriptors in a de facto-compact format that can be used in classification or recognition tasks. These operations are critical in embedded vision systems as the feature extraction part of the computational workload is a lot in many cases.

Upon extraction of relevant features, the processed data will be sent to the vision algorithm implementation phase where one will perform higher level analysis. In numerous current embedded vision systems, light convolutional neural networks (CNNs) are being involved in the activities of object recognition, image identification, and pattern recognition. CNN based techniques are quite accurate in visual recognition tasks but very delicate to optimise when used in embedded systems with limited resources. The feature-based or traditional classification techniques might also be used as an alternative to deep learning models in cases where it is necessary to have lower complexity of computations. These techniques make use of features that have been extracted to carry out classification by use of machine learning models or knowledge-based decision system. The architecture of vision processing in this work is aimed at providing effective implementation of such algorithms with the help of hardware acceleration and efficient dataflow organisations.

Embedded vision systems also have an important requirement of being able to process image streams on the fly. The availability of real-time processing is to make sure that visual data can be processed sufficiently fast to allow responsive decisions to be made in processes like autonomous navigation or in industrial inspection. To accomplish this goal, rather than buffering and delays between processing units that make up embedding vision pipelines, streaming architecture is commonly used where image data is continuously sent through these processing units. Under this streaming model, it is possible to run the several steps of a pipeline in parallel with parallel processing architectures that are executed on FPGA platforms. Computation can be overlapped at multiple points within the pipeline so the system is able to greatly decrease the processing latency and still provide high throughput. Data transfer, memory access and parallel processing units are thus important in the attainment of real-time behavior in embedded vision applications. These stages interact to create an effective visual processing workflow as shown by the pipeline structure as illustrated in Figure 1. The sensor interface captures image data then it pre-processes and converts the data into feature representations and finally analysed by Vision algorithms to produce a classification or recognition result. The proposed framework can be successfully used to support sustained real-time image analysis with minimal computational load and energy usage by structuring the embedded vision system in the form of a structured processing pipeline.

Hardware-Software Co-Design Methodology

The design of the embedded vision systems must carry out a design approach which concurrently produces the best computing algorithm as well as the hardware structure. Hardware software co-design entails an organised approach to the delivery of this goal by segregating system functionality to programmable processors and dedicated hardware accelerators. This design style in embedded vision systems allows designers to trade between flexibility and the computational capabilities and ensure very tight control over power consumption, latency, and resource usage. The suggested co-design model combines the optimised vision algorithms with the hardware accelerated via FPGA to provide the high throughput real time processing. The general architecture of the proposed hardware software co-design model is as depicted in Figure 2 that depicts the relationship among the embedded processor, FPGA accelerator modules, memory subsystem and the vision processing pipeline.

The co-design model brings together the use of software based control logic and hardware accelerated processing modules together to form an efficient embedded vision platform. The embedded processor manages the systems level in this architecture with its work in sensor control,

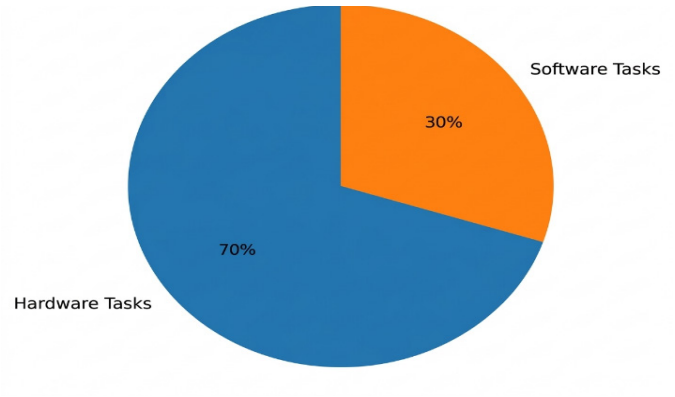

Fig. 2: Hardware–software co-design framework for FPGA-based embedded vision processing architecture.

configuration, algorithm coordination and decision level processing. At the same time computational intensive image processing functions are described in hardware modules contained in the FPGA field. The flexibility of software can be exploited and the parallel processing ability of FPGA hardware can be used with this integration. With this coordinated architecture, the system would be able to perform much better than when it is done with traditional processors implementations where it is not executed in a coordinated architecture. One of the most important steps in the co-design methodology situation is optimization of algorithm to map into hardware. Algorithms used to perform vision, which were originally designed to run on general-purpose processors, may even include inefficient operations to use them directly on hardware platforms. Thus, they need to restructure and simplify algorithms to facilitate the effective use of FPGA implementation. The loop unrolling technique, pipeline restructuring and algorithm decomposition techniques are frequently used in converting complex vision tasks to be presented as hardware-friendly designs. Besides these, computational kernels in the algorithm are also identified and optimised to run in parallel. The system is able to decrease processing latency and enhance a throughput by simplifying the algorithmic operation and restructuring the algorithm to get it to run on hardware.

Here task partitioning between hardware and software components is another significant design process in the co-design scheme. Efficient partitioning involves the choice of the operations to be performed by the embedded processor and those to be performed as hardware accelerators in the FPGA. The programmes that execute on the processor usually do control oriented tasks like system setting, decision logic as well as communication management. Conversely, computationally intensive algorithms which include image processing, convolution, feature detection, and neural network prediction are transferred to hardware modules associated with FPGAs. The partitioning strategy guarantees that FPGA hardware can be used to provide more time-sensitive operations with time-sensitive applications and the flexibility of the system with software-based control. The dataflow optimization is another important point of the proposed architecture to facilitate real-time image processing. Embedded vision systems frequently handle continuous data in terms of image frames and it is therefore necessary to move data efficiently in order to sustain a high throughput. The proposed design uses streaming data architecture to enable an image data flow to occur without unnecessary buffering or memory overhead between processing modules. The FPGA uses pipelined processing units that become active simultaneously on various images processing pipeline phases. Such streaming architecture reduces the delay of accessing memories and enables the execution of multiple processing stages concurrently which increases the performance of the system to a large extent.

The other aspect that is critical in the design of FPGA is strictness and optimization of resources. Hardware implementations should effectively consume fewer FPGA resources that include logic block, DSP, and on-chip memory. Use of fixed-point arithmetic as an alternative to floating-point computation is one of the successful methods to pursue this goal. Fixed-point representations use less hardware and faster arithmetic operations can be performed, albeit with adequate numerical accuracy to perform a wide range of vision processor tasks. Moreover, the neural network parameters and the in-between feature maps can have their precision reduced employing methods that will reduce the computational complexity even more. The proposed framework offers a certain level of performance, accuracy, and hardware efficiency by providing the optimal resource utilisation and choosing the right precision levels. In general, the co-design methodology of hardware and software as shown in Figure 2 offers a scalable and efficient system of deploying embedded vision systems on FPGA support systems. The proposed framework allows high-performance real-time image processing through optimising algorithms, task division, streaming dataflow architecture, and precision-aware hardware design at low-energy costs and flexibility of the system. The practise of co-design is relevant to support high-level embedded vision applications in the field of autonomous systems, smart surveillance, robotics, and intelligent internet of things equipment.

Proposed FPGA Architecture

The embedded vision system proposed is developed with a reconfigurable FPGA-based system with the ability to accomplish high-performance real-time image processing

at low energy cost and scalability. FPGA will be a perfect platform to implement embedded vision applications since it supports parallel processing, customization of hardware at a hardware level, and reconfiguration. Using FPGA hardware to run computationally intensive vision algorithms directly eliminates processing latency and has a much higher throughput than the processing-based implementations. It is a proposed architecture of an embedded vision system using FPGA that combines vision processing units, hierarchical memory system, and reconfigurable processing units in a hardware-software co-design. The system architecture of the entire system comprises an image acquisition interface, embedded control processor, FPGA based accelerator modules, and a hierarchal subsystem of memory. Camera sensor has an image frame capturing and transmitting capability connected to FPGA via high-speed interface. After the image data has been captured into the FPGA fabric it is processed by a series of hardware accelerator modules that are custom designed to perform vision processing functions. The in-built processor coordinates the system-level functions (i.e. configuration management, scheduling and communicating with external devices). This architecture enables the system to be hybrid by providing flexibility of software and the computing power of a dedicated hardware accelerator.

In the FPGA fabric, modules of image processing accelerators are realised in order to perform the main functions of the embedded vision pipeline. The feature extraction unit is one of the main units as it does functions like the determination of edges, filtering, and gradient to distinguish the pertinent patterns in the captured frames of images. These functions are carried out with parallel processing architecture so that large volume of data processing may be applied. A convolution engine is another important element, it executes convolution operations which are important in many vision algorithms, especially convolutional neural networks (CNNs). Convolution engine employs a number of parallel multiply-accumulate units in order to hastens the convolution calculations on image feature maps. The system, after the convolution operations, uses the activities of activation and pooling modules, which implement nonlinear transformations and spatial down-sampling of its feature maps. Activation functions make the processing pipeline nonlinear enabling the vision algorithm to learn complicated visual patterns. Pooling provides a reduction in the spatial resolution of feature maps with the retention of important features, and thus, consumes less memory and less calculation of feature maps. Pipelined versions of processing modules are used to make sure that the hardware accelerator is used in continuous data flow.

An effective information flow is realised by a well-considered memory hierarchy comprising of on-chip and off-chip memory devices. On-chip buffers On-chip buffers are realised by FPGA block RAM (BRAM) and are used to provide high-speed local storage of intermediate data (e.g. image tiles and feature maps). These buffers allow fast access speeds to hardware accelerator modules as well as minimise latency in terms of external access to memory. The architecture has also an external memory interface to off-chip DRAM modules when there are a large number of model parameters and large datasets. The external memory subsystem is used to store image frame contents which are huge, neural network weights, and processing results, which are not completely handled in on-chip memory resources. The architecture also has reconfigurable processing elements which help to compute in parallel multiple image processing functions. These processing units are modular computing units that perform arithmetic and logical functions needed to run vision algorithms. The system can generate these units in the FPGA fabric and thus multiple data components can be calculated at the same time enhancing the throughput dramatically. The FPGA hardware reconfigurability also enables these processing units to be tailored to various algorithms or workloads, such that a large variety of embedded vision systems can be supported by the system.

The dataflow design and the pipeline design are also another critical aspect of the proposed architecture since they allow processing of images through streaming. Image data is processed in a series of modules that forms a hardware line in this architecture. The stages of the pipeline presumably execute a particular operation, e.g., a filter, convolution, activation, or classification. Since several stages of the pipeline are running simultaneously, various parts of an image can be processed in different stages of the pipeline. This scale out pipelined streaming architecture is used to minimise the total processing latency and to optimise the use of hardware. The embedded architecture illustrates the cooperation of these elements in order to provide a functional embedded vision processing platform. This architecture is comprised of FPGA-based accelerator, optimised memory hierarchy, reconfigurable processing elements, and streaming dataflow architecture to allow the system to perform with high throughput, low latency, energy-efficient operation. The architecture offers an extensible platform of applying high-level embedded vision work in autonomous robotics, smart surveillance systems, intelligent transportation, and industrial inspection systems.

FPGA Deployment and Hardware Implementation

To test the capability of the proposed embedded vision architecture in reconfigurable FPGA platform, its real-time

processing capacities, hardware efficiency, and usability in embedded vision application were tested. Direct hardware acceleration of computationally complex functions in FPGA implementation, which can include convolution, feature extraction, and activation functions, is found to provide the system a much greater throughput and reduced latency when compared to software-based FPGA implementations. The deployment process entails the synthesis of the vision accelerator modules, combination of the modules with the embedded processor sub system and configuration of the memory interfaces to accomplish image streaming continuously and process data at high rate.

Target FPGA Platform

The hardware prototype was then deployed on a recent FPGA development board used in the application of embedded computing and real-time signal processing. The FPGA platform chosen offers enough logic processing units, on-chip memory blocks and digital signal processing units needed to execute a high throughput vision accelerator. It comprises of programmable logic fabric, embedded processing capability, and high-speed external memory interfaces, which make it useable in real time embedded vision systems. The FPGA resembles the small chip architectures found on other component-based systems; its contents, including configurable logic blocks, block RAM (BRAM), and DSP slices to perform arithmetic operations and programmable I/O interfaces to be connected to external sensors and memory modules, are all included in the chip. Direct acquisition of image frames is made possible by the camera interface, whereas large image datasets and neural network parameters are stored by the external DRAM interface. The integrated processor subsystem is in charge of system configuration, coordination of data flows, as well as communication between the FPGA hardware subsystem and external hardware.

Design Flow

The implementation and design of the FPGA vision accelerator is done in a hardware/software co-design approach. Vision processing pipeline was initially modelled on high level in terms of algorithmic descriptions of image processing operating units which include preprocessing, convolution, activation and pooling. They were then converted into hardware implementations by use of High-Level Synthesis (HLS) and Register Transfer Level (RTL) design languages. The high-level synthesis is used to simplify the process of converting high-level algorithmic specifications written in C/C++ into resulting hardware modules to facilitate prototyping hardware and exploring designs. Parallel processing dataflow structures and pipelined dataflow structures were optimised with critical processing components such as convolution engines and feature extraction modules to get maximum throughput. The hardware modules obtained were incorporated with RTL custom components that took control of the memory access, buffering, and streaming of data interfaces. An FPGA design tools was used to synthesise, place, and route the final hardware design to produce the required configuration bitstream to be loaded on the FPGA board. The resulting architecture is a streaming pipeline which receives image data and gives one an output which is continually read and processed to provide real-time embedded image processing.

Resource Utilization

The use of resource is another substantial indicator when assessing the hardware efficiency of accelerators based on FPGA. The enacted architecture incorporates the combinations of configurable logic blocks, flip-flops, digital signal processing units and block RAM resources, which are used to implement the embedded vision pipeline. Combinational logic of control units, routing of data, and arithmetic operations are implemented with the help of Lookup Tables (LUTs). Flip-flops are used in storing inter-stage processing pipeline states and storing inter-stage synchronisation signals. In the convolution engine, DSP blocks are extensively utilised in order to accomplish multiply-accumulate operations necessary to convolutional filtering and feature detection. Block RAM resources are on-chip high-speed storage resources, used by the accelerator modules to store intermediate feature maps, sliding window buffers as well as transient data. The hardware resource usage of the devised FPGA vision accelerator can be detailed by Table 2 that shows the allocation of the logic components and memory unit to the hardware modules. The findings reveal that the suggested architecture is easy to use without over consuming the FPGA resources and has enough scalability to accommodate the more complicated vision workloads.

Table 2: FPGA resource utilization of the proposed embedded vision accelerator

Resource Type	Utilization	Percentage of Device
LUTs	32,480	45%
Flip-Flops	41,210	38%
DSP Blocks	120	50%
BRAM	96	42%

Clock Frequency and Timing Analysis

The frequency and timing performance of the clocks are critical metrics in measuring the real-time processing capability of the FPGA based systems. The architecture proposed was such that pipelined processing stages were used to make sure that several operations of processing images take place at the same time. This pipeline design

means that idle cycles are reduced to a minimum and the system throughput becomes better. The synthesis and implementation tools of FPGA were used to run timing analysis to ensure that all important timing constraints had been met. The findings suggest that the system can stabilise at an operating frequency that can be used in real-time embedded vision. The convolution engine along with memory buffering architecture is optimised to enable the system to maintain continuous image streaming without any timing violation. The timing parameters and clock Compensation of the FPGA accelerator are presented in Figure 3 that demonstrates the timing distribution and clock domain interactions of the processing pipeline. The findings prove that the architecture has reliable timing contents and can support parallel processing with huge throughput. In general, the hardware development shows that the suggested FPGA-based vision accelerator has the potential for an efficient resource utilisation, predictable timing behaviour, and scalable deployment functionality of real-time embedded vision systems.

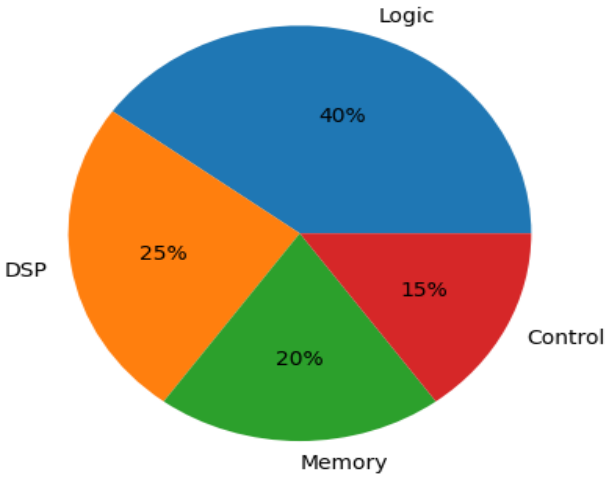

Fig. 3: FPGA Resource Distribution of the Proposed Vision Accelerator Architecture.

Experimental Results and Performance Evaluation

To measure the occurrence of real-time performance, computational efficiency and energy consumption of the proposed FPGA-based embedded vision architecture performance was carried out experimentally. The consideration is processed latency, system throughput and power efficiency that is important parameter in real-time embedded vision system. The common CPU-based and the GPU-based processing systems, and other previously implemented vision accelerators based on the FPGA was also compared to the performance of the proposed architecture and was used to demonstrate its benefits in embedded system implementation.

Experimental Setup

The proposed experimental system is composed of a FPGA-based embedded vision platform that is connected with a camera input subsystem and external memory subsystem. The camera sensor uses continuous image frame capture which is streamed to the FPGA fabric via a high speed interface. The input set consists of a collection of regular image frames that are usually taken to test the computer vision algorithms, such as scenarios of object detection and classification. The input frames are processed in the pipeline of FPGA accelerator and run at the preprocessing, convolution, activation parts and pooling parts. The integrated processor controls the configuration of the system and the transfer of the information between that of the camera interface, accelerator modules, and the external memory. The processing latency, frame throughput and energy consumption were taken during the continuous operation of the embedded vision pipeline.

Latency Analysis

Latency is a decisive factor when real-time embedded vision systems are concerned since it establishes the speed at which the system can process incoming image frames. The presented FPGA architecture is based on the pipelined streaming dataflow design, which enables the example of the vision processing pipeline to run several steps at the same time. The fact that frame images are processed streamingly makes the effective processing delay of each frame much lower than that of sequential processing methods of CPU-based systems. The convolution engine, as well as parallel feature extraction modules, have been optimised to enable the system to work with the image frames with minimum delay. The latency character of the proposed architecture is shown in Figure 4 which represents the processing time per frame needed in the FPGA-based accelerator versus CPU and GPU implementations. The findings indicate that the suggested FPGA architecture can attain significantly reduced processing latency values because of its parallel hardware acceleration and pipelined architecture.

Throughput Evaluation

Throughput is the rate of image frames per second and useful in the assessment of scalability of embedded vision systems. The processing pipeline formed by streaming that is applied to the FPGA allows running continuous image processing without requiring the completion of intermediate processing steps. The convolution engine uses multiple parallel multiply accrue units with which many pixels and feature maps can be processed at the same time. Consequently, recognition of computationally intense vision tasks in the system is attained at high

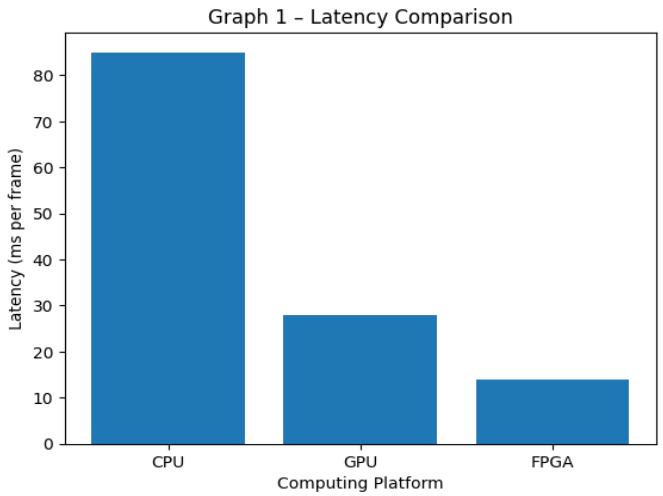

Fig. 4: Latency Comparison of CPU, GPU, and FPGA Platforms for Real-Time Vision Processing.

frame processing rates. Figure 5 shows the throughput evaluation results of the proposed FPGA accelerator in terms of comparing the performance in respect of frames-per-second (FPS) of the proposed FPGA accelerator and CPU-based and GPU-based vision processing systems. The findings show that FPGA implementation has much higher throughput than CPU-based systems with a competitive performance as compared to the implementation that uses the GPU and has lower power consumption by a significant percentage margin.

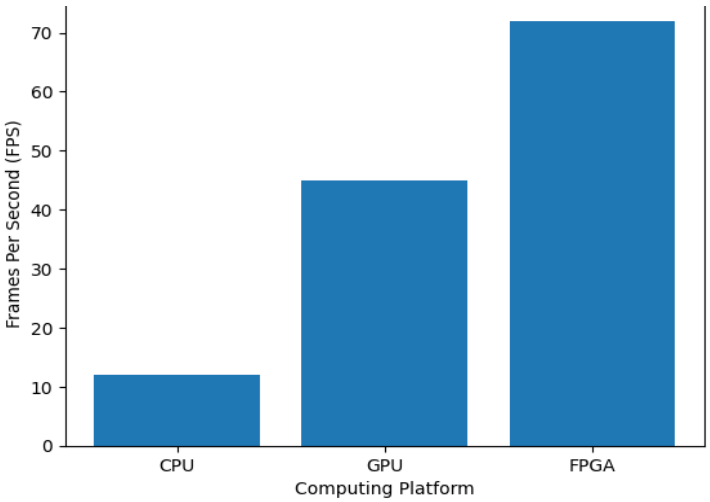

Fig. 5: Throughput Performance Comparison of CPU, GPU, and FPGA Platforms in Frames per Second (FPS).

Power Consumption

FPGA-based embedded vision systems have a great benefit in energy efficiency. In contrast to systems based on GPUs which utilise large populations of general-purpose processing cores, FPGA accelerators are specialised-purpose hardware recreational centres that are optimised to carry out specific computational tasks. The power consumption of the proposed architecture was observed when there was a continuous operation of processing images. The findings have shown that the FPGA accelerator uses a lot less power and still achieves a high computational throughput. Comparison in power consumption of various computing platform is demonstrated in Figure 6 which depicts energy efficiency of CPU, GPU, and FPGA implementations. The findings indicate that the designed FPGA architecture is more energy efficient and thus has high potential to be applied to these forms of power-constrained embedded systems which include autonomous robots, surveillance systems, and industrial monitoring systems among others.

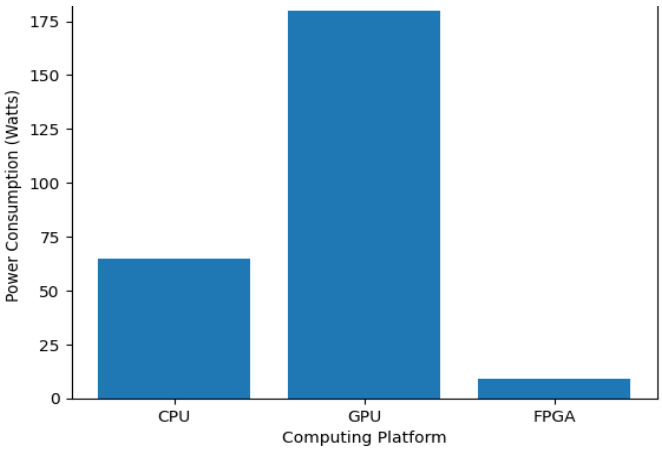

Fig. 6: Power Consumption Comparison of CPU, GPU, and FPGA Platforms for Embedded Vision Processing.

Comparison with Existing Methods

In order to measure the performance of the suggested architecture further, the system performance was contrasted with the current vision processing platforms, such as CPU-based implementation, GPU-based accelerators, and previously reported vision system based on FPGA. Implementations based on the CPU are more likely to run these vision algorithms in a linear fashion and therefore introduce a significant processing latency and a reduced throughput in real-time applications. The advantage of GPU systems is their performance due to massively parallel processing units but it typically takes much more power and has higher thermal management needs. In contrast, the presented FPGA architecture ensures a so-called balanced approach to the design of hardware acceleration with the energy efficiency of the functioning. The architecture employs the optimised convolution engines, pipelined processing modules and sensible management of memory to give excellent

Table 3: Performance comparison of the proposed FPGA vision accelerator with existing computing platforms

Platform	Latency (ms/frame)	Throughput (FPS)	Power Consumption (W)
CPU-Based System	85	12	65
GPU-Based System	28	45	180
Existing FPGA Architecture	22	50	18
Proposed FPGA Architecture	**14**	**72**	**9**

embedded vision processing. Table 3, gives a comparative overview of the performance of various systems in diverse platforms and clearly shows the gains in the latency, throughput, and power efficiency of the proposed FPGA-based architecture.

All in all, experimental findings indicate that the proposed FPGA-based embedded vision architecture is able to make major gains in processing speed, throughput, and energy efficiency. Parallel Hardware acceleration combined with pipelined dataflow architecture and optimised memory hierarchy allows the system to meet the extreme needs of real time embedded vision application.

Discussion

The experiment findings indicate that the proposed FPGA-based embedded vision platform offers major gains in processing capabilities, reduction in the latency, and energy efficiency relative to traditional computing platforms. By applying computationally intensive vision functions like convolution, feature extraction, and activation functions directly in FPGA hardware, it is possible to achieve very parallel execution and, therefore, significantly reduce the processing time per frame. As seen in the performance analysis, the FPGA accelerator will have low latency and throughput and much lower power usage than CPU-based implementation as well as the much lower power usage of the system compared to the GPSU-based systems. These advances underscore the efficiency of the use of hardware acceleration to take care of the compute issue of real-time embedded vision applications. Hardware-software co-design is one of the key features that the proposed architecture will offer. Under this concept, the embedded processor will handle the high-level system control, configuration management and schedule tasks whereas FPGA fabric will implement time critical vision processing tasks using specified hardware accelerators. Such partitioning of duties enables the system to give a combination of the flexibility of the software with the high-computational performance of the specialised hardware. Consequently, the system will be able to cope with varying loads of vision work and various algorithm parameters without negating its performance. The co-design approach makes system integration simpler as well as allows effective communication among software control subunits and hardware accelerator units.

The other critical thing that can be stated about the proposed architecture is its scalability to embedded vision systems in the future. The FPGA accelerator is a modular design which means that there can be an addition of more processing units to the architecture as the needs of computational requirements grow. Since FPGA platforms have reconfigurable logic units, processing elements and convolution units can be increased or decreased as required. This allows the architecture to be reconfigured to execute more complex vision algorithms, such as the more complex deep learning models and multi-stage image processing pipelines. Moreover, the streaming pipeline layout will enable addition of processing stages without changing the overall layout of the system greatly. The optimised hierarchy of memory and parallel processing architecture are also factors which make the architecture scalable. On- chip buffers minimise access latency of data and external DRAM interfaces enable the system to utilise bigger datasets and neural network parameters. Such a mix makes it possible to assure that the architecture is able to scale to more image resolution, model complexity, and processing demands. To this end, the suggested FPGA-incorporated framework is a powerful and scalable framework of next-generation embedded vision in areas like autonomous robotics, smart surveillance, smart transportation framework, and automation within business industries.

Limitations

Although a boost in performance, especially with respect to processing power, is projected to be realised with the suggested FPGA-based embedded vision architecture, there are multiple limitations that need to be addressed. A lack of availability of FPGA hardware resources is one of the main limitations. The number of logic elements, DSP blocks and block RAM resources in FPGA resources are limited, and this limits the size and complexity of vision algorithms that can be directly implemented in hardware. Considering that the complexity of the modern computer vision models grows rapidly, especially with deep neural networks with significant volumes of parameters and layers, the issue of resource allocation becomes very essential. To ensure that a system operates at its best and not to saturate available resources, it is thus needed to

map the computational tasks effectively to the available FPGA resources. The other constraint is associated with the complexity of sophisticated vision models. Although the offered architecture effectively promotes convolution-based feature extraction and schemes of a straightforward neural network, to undertake very deep or huge-scale model development, it might need extra hardware streamlining methods, including model compression, quantization or layer reconfiguration. Such optimizations are needed to overcome the on-chip memory and computational power available in the FPGA device of large neural network architectures. As a result, the models of the high complexity might have to be partitioned in addition to several hardware devices or utilise external computing capabilities.

Additionally, the present evaluation is mostly based on the prototype implementation and performance measurements done with FPGA on the controlled experimental setup. Even though the outcomes reveal good prospects of embedded vision processing in real time, there is still a need to do more hardware testing to completely affirm the reliability and security of the system in the real world. Applied deployment conditions, e.g. autonomous systems or industrial monitoring platforms will cause more constraints such as environmental variability, sensor noise, and communication latency. The work in the future should therefore involve large scale hardware testing, system integration and long term performance validation to make sure that the applicability of the proposed architecture in a wide range of embedded vision environments is practical.

Future Research Directions

The proposed FPGA-based embedded vision architecture can be further promoted to provide better capabilities and applications in future research by incorporating more common hardware acceleration methods and smart processing models. The addition of special deep learning accelerators to the FPGA board is one of the directions. Since deep neural networks are becoming more commonly used in glaucoma studies in modern vision applications, the computational efficiency of these applications can be substantially enhanced by adding specialised neural network inference hardware. Such convolutional neural networks, quantized neural networks and lightweight deep learning models can be added to the existing processing pipeline to enable more sophisticated visual recognition and classification and at the same time remain visible in real-time. The other potential area of research is the implementation of edge AI vision systems that operate according to the presented architecture. Edge computing facilitates the data processing process to be done at the sensing device as opposed to cloud infrastructure that is centralised. Combining the hardware acceleration of FPGAs with edge AI algorithms, embedded vision systems are capable of making intelligent decisions and doing this locally with minimum latency and reduced communication overheads. These systems would be especially useful in systems like smart surveillance, industrial automation, health monitoring, and intelligent transportation where fast and accurate visual inspection is a crucial requirement.

It is also suggested that dynamic partial reconfiguration of FPGA hardware is another relevant field in future research. It is possible with this capability to reconfigure certain parts of the FPGA fabric at runtime without affecting the functionality of the rest of the system. Different vision processing modules or layers of a neural network could be loaded on demand by making use of the dynamic reconfiguration process based on the need as per the application requirement. This would make use of hardware resources efficient and the embedded vision system dynamic to different workloads and algorithmic demand. Lastly, the work should be conducted in the future through the implementation of the proposed architecture in autonomous and real independent intelligent systems. Robots, autonomous flying vehicles, driver-assistance systems, and intelligent manufacturing systems cannot be done without high-performance real-time vision processing with strong limitations on power use and computation time. Application of the suggested FPGA-based architecture in such systems would enable researchers to test the performance of the proposed system in realistic operating conditions and further optimise the system design towards large-scale systems implementation. The embedded vision architectures with FPGA can be a major contributor to realising next generation intelligent sensing and autonomous systems through these developments.

Conclusion

This paper has introduced a hardware-software co-design architecture in order to deploy a high-performance embedded vision system on a programmable FPGA-based architecture. The proposed architecture incorporates an embedded processor and Hardware accelerators in the form of FPGAs to execute computational intensive vision processing functions like preprocessing, convolution, activation and pooling functions efficiently. The architecture makes costly use of the built-in parallelism and pipelined processing of the underlying FPGA hardware to achieve large improvements in processing latency, throughput and energy efficiency over traditional CPU-based and GPU-based systems. The experimental

analysis shows that the optimised memory hierarchy, the streaming dataflow architecture and the dedicated convolution engine are able to provide real time image processing, which can be used in embedded vision applications. These findings demonstrate the appeal of reconfigurable architectures based on FPGA as a potential platform in intelligent embedded systems of the future, which has scalability, adaptability, and high computational throughput, such as a next-generation edge AI platform and autonomous vision platform.

References

1. Chen, Y. H., Krishna, T., Emer, J. S., & Sze, V. (2016). Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks. *IEEE journal of solid-state circuits*, *52*(1), 127-138.
2. Cong, J., & Xiao, B. (2014, September). Minimizing computation in convolutional neural networks. In *International conference on artificial neural networks* (pp. 281-290). Cham: Springer International Publishing.
3. Giri, D., Chiu, K. L., Di Guglielmo, G., Mantovani, P., & Carloni, L. P. (2020). ESP4ML: Platform-based design of systems-on-chip for embedded machine learning. *arXiv preprint arXiv:2004.03640*.
4. Gupta, S., Agrawal, A., Gopalakrishnan, K., & Narayanan, P. (2015, June). Deep learning with limited numerical precision. In *International conference on machine learning* (pp. 1737-1746). PMLR.
5. Han, S., Mao, H., & Dally, W. J. (2015). Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*.
6. Haris, J., Gibson, P., Cano, J., Agostini, N. B., & Kaeli, D. (2021, October). SECDA: Efficient hardware/software co-design of FPGA-based DNN accelerators for edge inference. In *2021 IEEE 33rd International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD)* (pp. 33-43). IEEE.
7. Hegarty, J., Eldash, O., Suleiman, A., & Alaghi, A. (2021). HWTool: Fully Automatic Mapping of an Extensible C++ Image Processing Language to Hardware. *arXiv preprint arXiv:2110.12106*.
8. Hennessy, J. L., & Patterson, D. A. (2011). *Computer architecture: a quantitative approach*. Elsevier.
9. Huang, Q., Wang, D., Gao, Y., Cai, Y., Dong, Z., Wu, B., & Wawrzynek, J. (2019, December). Algorithm-hardware co-design for deformable convolution. In *2019 Fifth Workshop on Energy Efficient Machine Learning and Cognitive Computing-NeurIPS Edition (EMC2-NIPS)* (pp. 48-51). IEEE.
10. Kryjak, T. (2024, August). Event-based vision on fpgas-a survey. In *2024 27th euromicro conference on digital system Design (DSD)* (pp. 541-550). IEEE.
11. LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *nature*, *521*(7553), 436-444.
12. O'shea, T., & Hoydis, J. (2017). An introduction to deep learning for the physical layer. *IEEE Transactions on Cognitive Communications and Networking*, *3*(4), 563-575.
13. Ogundairo, O., Anggreani, C., Situ, L., & Zhang, Y. (2025). A Practical Guide to Hardware-Software Co-design for Real-Time Image Processing Pipelines.
14. Simeone, O. (2018). A very brief introduction to machine learning with applications to communication systems. *IEEE Transactions on Cognitive Communications and Networking*, *4*(4), 648-664.
15. Yamini, V., Hussain, S. A., Chandra Sekhar, G., Avinash Kumar, P., Lehitha, P., Sree Venkata Teja, B., ... & Sanki, P. K. (2024). An SoC system for real-time edge detection. *Journal of Electronic Materials*, *53*(10), 6395-6402.
16. Yan, Z., Zhang, B., & Wang, D. (2024). An FPGA-based YOLOv5 accelerator for real-time industrial vision applications. *Micromachines*, *15*(9), 1164.