



# An SoC System for Real-Time Edge Detection

Vanama Yamini<sup>1</sup> · Syed Ali Hussain<sup>1</sup> · G. Chandra Sekhar<sup>1</sup> · P. Avinash Kumar<sup>1</sup> · P. Lehitha<sup>1</sup> · B. Sree Venkata Teja<sup>1</sup> · Swagata Samanta<sup>1</sup> · Pradyut Kumar Sanki<sup>1</sup>

Received: 30 September 2023 / Accepted: 7 June 2024 / Published online: 3 July 2024  
© The Minerals, Metals & Materials Society 2024

## Abstract

This research work focuses on the design and implementation of a highly advanced field-programmable gate array (FPGA)-based system-on-chip (SoC) solution for real-time edge detection. By utilizing a Zynq processor and leveraging the powerful Vivado software, the aim is to overcome the significant computational challenges associated with achieving real-time edge detection. Edge detection in real-time scenarios presents several obstacles, including the possibility of missing edges due to noise and the substantial processing requirements of any edge detection technique. To address these challenges, the proposed SoC system synergistically combines the computational capabilities of an FPGA board and a Zynq processor, harnessing hardware acceleration to achieve high-performance edge detection. The OV7670 camera module serves as the primary input medium, capturing image frames for subsequent processing. These captured frames undergo initial processing before being seamlessly transferred to the FPGA fabric through customized intellectual property (IP) blocks. These IP blocks efficiently handle crucial tasks such as frame capturing, conversion to AXI Stream interface signals, and integration with the video direct memory access (VDMA) IP. The VDMA IP plays a pivotal role by facilitating high-speed data movement between the FPGA fabric and the Zynq processor IP, thereby enabling streamlined and efficient data transfer and processing. At the heart of this project lies the real-time edge detection algorithm, which is skillfully implemented on the Zynq processor. The resulting edge-detected frames are then visually presented and displayed on an output device utilizing the AXI4-Stream to Video Out IP. To ensure optimal utilization of available hardware resources, the comprehensive Vivado software suite provides a wide array of tools for designing, implementing, and programming the FPGA fabric. By leveraging FPGA-based systems, this project effectively addresses the critical need for real-time edge detection in time-sensitive scenarios. The result is a portable and manageable device that exhibits versatility, as it can be employed in various applications while reliably detecting edges in real-time situations.

**Keywords** FPGA · SoC · Zynq processor · Computer vision · Image processing · Hardware acceleration

---

✉ Pradyut Kumar Sanki  
pradyut.s@srmmap.edu.in

Vanama Yamini  
yamini.va@srmmap.edu.in

Syed Ali Hussain  
alihussain\_syed@srmmap.edu.in

G. Chandra Sekhar  
chetanvdsngoli@srmmap.edu.in

P. Avinash Kumar  
avinashananda@srmmap.edu.in

P. Lehitha  
lehitharama@srmmap.edu.in

B. Sree Venkata Teja  
sreevenkatatejabollina@srmmap.edu.in

Swagata Samanta  
swagata.s@srmmap.edu.in

<sup>1</sup> Department of Electronics and Communication Engineering, SRM University-AP, Amaravati 522502, India

## Introduction

Detecting edges in images is essentially the initial step for in-depth image analysis. This vital method improves image quality and helps with a variety of analytic tasks. When it comes to increasing the processing speed of image processing systems, Ganesan and Sajiv<sup>1</sup> noted that hardware implementation is crucial. Essentially, our goal in discussing edge detection implementation on hardware such as the Edge Z7-10 board is to achieve a considerable improvement in image processing speed. This board offers an effective framework for creating real-time edge detection systems that work well by combining a field-programmable gate array (FPGA), a Zynq central processing unit (CPU), and Vivado software.<sup>2</sup> In real-time circumstances, our goal is to create a system that can quickly and efficiently identify edges by utilizing these hardware capabilities. The processing speed must be extremely fast due to the large amount of image data that must be processed to enable image data transmission in real time.<sup>3</sup> The operating time of the image processing module is shortened by the median filter, Sobel edge detection method, and corrosion expansion algorithm when combined with the real-time edge detection algorithm for picture data with  $1280 \times 720$  resolution and 30 frames per second (fps). Many embedded systems, where fast execution is significant, call for real-time edge detection.<sup>4</sup> Embedded multiprocessor systems provide a practical solution for meeting the real-time demands of computationally intensive applications, particularly those that are highly parallelizable.<sup>5</sup> In our research, we proposed a co-design approach for multiprocessor system-on-chip (MPSoC) hardware and software tailored for real-time Sobel edge detection. This approach effectively integrates FPGA and IP blocks. Additionally, we focus on implementing IP blocks necessary for this application. Edge detection represents the initial step in image and video processing. The edges identified serve as valuable inputs for higher-level tasks such as image enhancement, object recognition, and object tracking, facilitating a seamless flow in the processing pipeline.

An extremely quick system is needed to complete this operation in real time. The software does not appear to be a good choice for real-time implementation. We require technology with a high degree of parallelism. As was noted, we need a good platform for real-time implementation, so we have chosen Vivado for design. The edge detection hardware includes a peripheral that is a 32-bit soft RISC TSK3000A. One of the essential aspects of the image is edge detection.<sup>6</sup> It serves as a foundation for many aspects of image analysis, including target area identification, extraction, and other regional forms. The vast variety of edges in a picture are detected by edge

detection operators such as the Canny, Sobel, and Prewitt operators. To create a gradient image, these operators perform the convolution process on every pixel. Complex convolutions are simple to implement because of several intrinsic multipliers included in FPGA hardware components. These resources are conveniently employed in this design process, which promotes effective implementation. Reliability is maintained while consuming less memory, latency, and mathematical complexity due to algorithm simplifications. Electronic integrated circuits based on a matrix of reconfigurable logic blocks (CLBs) are known as FPGAs.<sup>7</sup> Traditionally, hardware description languages (HDLs) have been used to set up FPGAs.

In this paper, the implementation of edge detection on a real-time image capture is presented.<sup>8</sup> The edge-detected image is displayed in a video graphics array (VGA) on the Terasic 5 megapixel (MP) camera once the video has been recorded and interfaced with the FPGA kit. The OV7670 camera is a popular and widely used image sensor module known for its affordability and accessibility. The module also supports multiple output formats, including raw RGB, YUV, and compressed JPEG, providing flexibility in data processing and transmission. With its compact size, ease of integration, and reasonable image quality, the OV7670 camera module has found application in areas such as robotics, surveillance systems, image processing, and embedded vision projects. Additionally, the edge-detected version of the RGB image taken by the interfaced camera is obtained using the Canny edge detection technique.<sup>9,10</sup> In comparison with other edge detection algorithms, this one has a low error rate. Edge detection is one of the core techniques used in object recognition and image processing.<sup>11</sup> Due to its significance in image processing, the architecture must be accurate, with low latency. An adaptive threshold edge detection approach is created since the conventional Sobel edge detection system is unable to meet the requirement for precision.<sup>12</sup> The detection system consists of a complementary metal-oxide-semiconductor (CMOS) image sensor, an FPGA, and DDR2, among other components. High-performance Cyclone IV GX series FPGA and DDR2 enable good real-time performance. The experiment demonstrates that the Sobel adaptive threshold-based edge detection technique may be implemented on an FPGA. Additionally, the system exhibits great real-time performance and high precision.<sup>13</sup> The goal is the same, but it has been approached in different ways including sensors and DDR2 and so on. This architecture obviously provides benefits in the areas of energy consumption and performance dependability despite noisy images. The proposed system finds application in various domains, including autonomous vehicles, surveillance systems, robotics, medical imaging, and remote sensing. These applications benefit from accurate and real-time edge detection, enabling improved decision-making, object

recognition, and environmental analysis. Key contributions of the proposed work include the following:

1. Using the Canny approach at the block level addresses problems such as an abundance of edges in smooth areas and a lack of essential edges in detailed sections while retaining edge detection accuracy.
2. Adaptive threshold computation enhances accuracy by dynamically updating thresholds based on block characteristics and local curve distribution.
3. A novel approach for estimating block-based hysteresis thresholds using a nonuniform gradient magnitude histogram is introduced to enhance edge detection accuracy.
4. Integration with current codecs and low latency allow for quick edge identification in high-resolution media. Furthermore, the proposed approach outperforms earlier frame-based algorithms, particularly in noisy visual circumstances, as demonstrated by subjective and quantitative assessments.

The remainder of the paper is structured as follows: section II provides a comprehensive overview of the methodology, section III delves into the IPs utilized, section IV demonstrates the results and analysis, and the culmination of the paper is found in section V, which presents the conclusion and future scope.

## Methodology

The project focuses on utilizing an FPGA-based SoC architecture. FPGAs offer parallel processing capabilities, allowing for the acceleration of computationally intensive tasks. The Edge Z7-10 board, which incorporates a Zynq processor, provides a suitable platform for developing hardware-accelerated solutions. The core of the project lies in the real-time edge detection algorithm, which is implemented on the Zynq processor. By integrating the Canny edge detection algorithm, the system analyzes the input frames and accurately extracts edges. The resulting edge-detected frames are visualized and displayed on an output device via the AXI4-Stream to Video Out IP. At the beginning of this work, we have used Vivado software to create a new project that enables us to experiment with various tools.

## Vivado

Vivado is a powerful and comprehensive software development environment provided by Xilinx, a leading provider of programmable logic devices (FPGAs) and adaptive compute acceleration platforms (ACAPs). Vivado is continually updated with new features, enhancements, and device support to keep pace with the evolving needs of FPGA and

ACAP designers. It empowers designers to unleash the full potential of Xilinx devices and deliver highly optimized and innovative digital systems.

Key features and capabilities include design entry, design analysis and verification, IP integration, high-level synthesis (HLS), implementation, optimization, debugging, validation, partial reconfiguration, and system-level integration with its comprehensive toolset, intuitive interface, and extensive documentation and support resources. Vivado has become a widely adopted software platform for FPGA and ACAP development that enables designers to achieve faster time to market, higher performance, and greater design productivity. After starting a register transfer level (RTL) project in Vivado, we must choose the ideal board file that meets the project's requirements. We have used the Edge Z7-10 (or Edge Z7-20) as the main board. This opens a Vivado Flow Navigator, which provides us with the tools in hand to create new things. We generated the block design for our project using the Create Block Design feature. The primary IP that we have added is the ZYNQ Processing System.

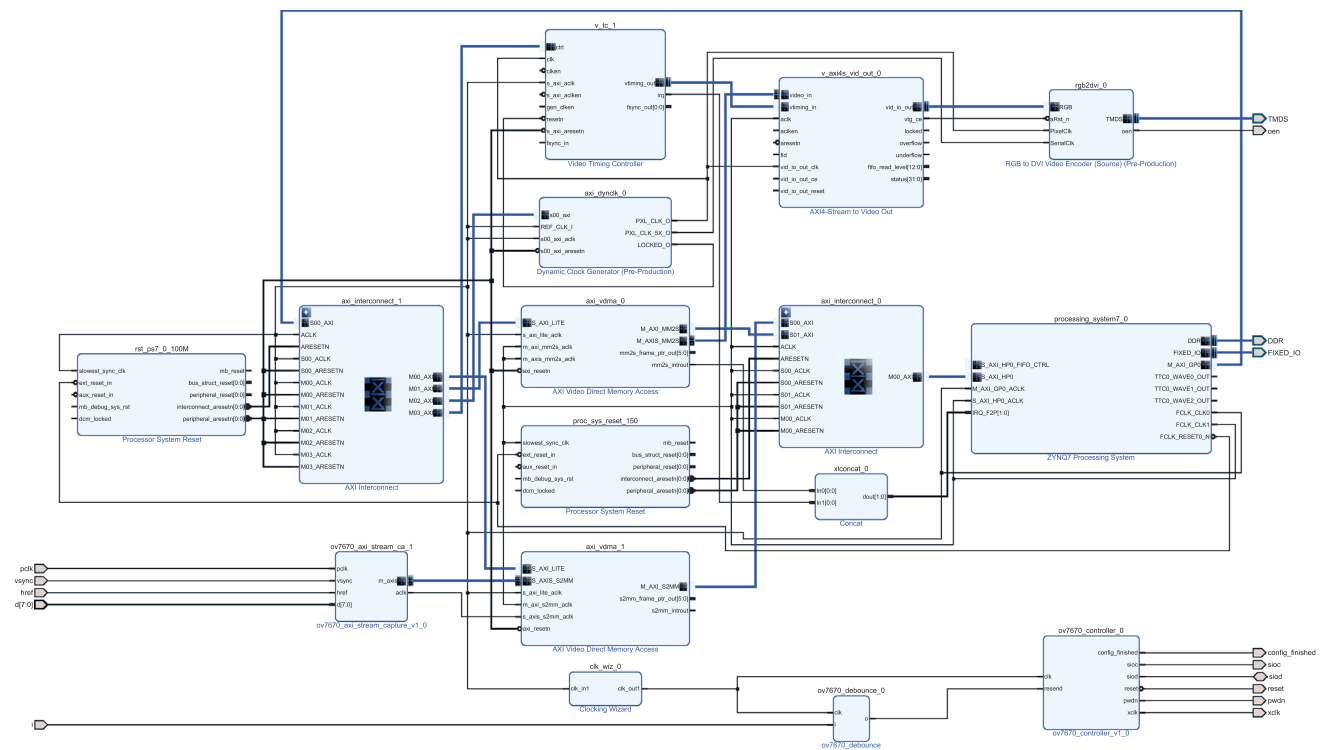
## IPs Utilized

### Zynq-7000

The Zynq-7000 Processing System (PS) IP is a fundamental component in the Vivado design suite provided by Xilinx. It is specifically designed to enable seamless integration of the processing capabilities of the ARM Cortex-A9 processor with the programmable logic fabric of Xilinx Zynq-7000 SoC devices. It serves as the foundation for creating advanced embedded systems and enables designers to leverage the benefits of both software and hardware in a unified platform. The Zynq-7000 PS IP includes a wide range of features and capabilities to facilitate the development of complex embedded systems. Key features and capabilities of the Zynq-7000 PS IP include the ARM Cortex-A9 processor, memory interfaces, system interconnect, peripherals and interfaces, interrupt controller, clock and reset management, debugging, and trace interfaces. We then integrate it with the AXI Video Direct Memory Access IP, Video Timing controller, AXI4-Stream to video out, AXI Interconnect, and other IPs, as can be clearly observed in Fig. 1.

### Video Direct Memory Access (VDMA) IP

The AXI Video Direct Memory Access (VDMA) IP is a key component in the Vivado design suite provided by Xilinx. It is a versatile IP block that facilitates the efficient transfer of video data between different modules within a digital system. The AXI VDMA IP is specifically designed to handle high-bandwidth video streams



**Fig. 1** Proposed IP block design for video capture.

and supports various video formats and resolutions. The AXI VDMA IP is widely used in video processing and display applications, including video surveillance systems, video streaming, image processing, and real-time video processing. It simplifies the design and implementation of video processing pipelines by efficiently managing the transfer of video data and offloading the processor from handling data movement tasks.

### AXI Interconnect IP

The purpose of the AXI Interconnect IP in this project is to serve as a central communication hub or switch-board, facilitating seamless connection and data exchange between various IPs along with the modules within the system. The AXI Interconnect IP acts as a bridge, ensuring proper interfacing and communication between the Zynq processing system, the custom IPs, and the peripherals. By incorporating the AXI Interconnect IP in the project design, the system can effectively manage the communication and data flow between different components, including the Zynq processing system, custom IPs, and peripherals. It enhances system reliability, scalability, and performance, allowing for seamless interaction and data exchange between various modules within the system.

### AXI4-Stream to Video Out IP

This IP can perform various operations such as video format conversion, AXI4-Stream compatibility, timing synchronization, and integration with the Video Timing Controller from the FPGA fabric with the video output interface. The AXI4-Stream to Video Out IP in Vivado plays a crucial role in bridging the gap between the processed video data in the FPGA fabric and the video output interface. The smooth presentation of the processed video data on the output display or device is made possible by ensuring compatibility, timing correctness, and simplification of the integration process.

### Video Timing Controller

The Video Timing Controller IP is utilized to provide precise control and synchronization of the video timing signals required for the proper display of the processed video data. It generates the necessary timing signals, controls the video resolution and refresh rate, synchronizes with the input video, and interfaces with the display controllers or output interfaces. By providing precise timing control, the IP ensures proper synchronization and display of the processed video data on the output device, resulting in high-quality and

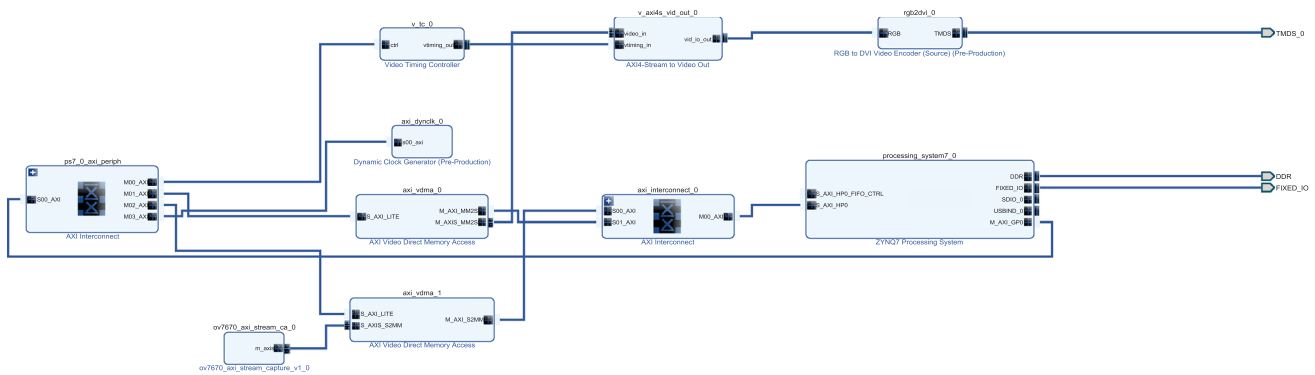


Fig. 2 Design flow of video timing controller.

visually appealing video. The design flow of the architecture is shown in Fig. 2.

### Internal Flow of the Model

This is a brief explanation of the internal process that produces the edge detection output. (1) **Input Media:** The OV7670 camera module is used as the input media for capturing images. The camera captures frames that contain visual information, which will be processed for edge detection. (2) **Initial Processing:** The captured frames undergo initial processing to enhance the quality of the images and prepare them for edge detection. This processing stage may include operations such as noise reduction, color space conversion, and image enhancement techniques. (3) **Custom IP Blocks:** Custom IP blocks are developed and integrated into the system to handle the image data. These IP blocks capture and record the frames sent by the OV7670 camera and convert the signals into a format compatible with the AXI-Stream interface. The custom IP blocks play a crucial role in preparing the data for further processing. These IP blocks are responsible for capturing and recording the frames, converting signals to the AXI-Stream interface format, and integrating with the VDMA IP. By developing custom IP blocks, the system can efficiently handle the data flow and prepare the frames for further processing. (4) **VDMA (Video Direct Memory Access):** The processed frames are transferred to the Zynq processor using the VDMA IP, which facilitates high-speed data movement between the FPGA fabric and the processor. The VDMA IP utilizes the AXI Stream protocol to efficiently transfer the processed image frames to the Zynq processor for subsequent analysis. This enables seamless communication and data flow within the system. (5) **AXI Interconnect IP:** The AXI Interconnect IP is used to establish communication and data flow between various IP blocks, including the VDMA IP, and other components in the system. It ensures smooth data transfer and proper coordination between different modules. (6) **Memory Management:** This

efficiently manages the memory resources to ensure smooth data transfer and processing. It sets aside the proper amount of memory buffers to hold gradients, edge data, and picture frames. The AXI4-Stream and VDMA IP interfaces can be used to read and write video data to memory. (7) **Timing and Synchronization:** The timing constraints and synchronization requirements to ensure proper coordination between the different components are considered. This includes synchronizing the input frames from the camera, coordinating the transfer of data between modules, and synchronizing the output frames for display. (8) **Testing and Debugging:** The implementation is thoroughly tested to validate its correctness and performance. Test images or video sequences are used to verify the edge detection results. Debugging of any issues or errors that may arise during testing ensures that the algorithm functions as expected. (9) **Edge Detection Algorithm:** Once the processed image frames are available in the Zynq processor, a real-time edge detection algorithm is applied. The specific edge detection algorithm used can vary based on the project requirements, but popular techniques such as Canny, Sobel, or Laplacian edge detection can be employed. The Canny edge detection algorithm is implemented in a programming language that is compatible with the Zynq processor, such as C/C++. (10) **Performance Optimization:** The algorithm parameters and hardware configurations are fine-tuned to optimize the performance of the real-time edge detection system. This may involve adjusting the smoothing kernel size and threshold values, or exploring parallelization techniques to fully leverage the capabilities of the FPGA fabric. (11) **Output and Display:** The resulting edge-detected frames are visualized and displayed on an output device. This involves converting the processed frames to a format compatible with the AXI4-Stream to Video Out IP, which connects to the output display, such as a High-Definition Multimedia Interface (HDMI) monitor. The visualization of the edge-detected frames enables real-time analysis and evaluation of the algorithm's performance.



## Process

After connecting all the required IPs together, we validate the design and then select Create HDL Wrapper. This creates a hierarchy with the wrapper on the top. Next, we generate output products. We proceed with Generate Bitstream. Generating a bitstream is a crucial step in the FPGA design flow as it produces a binary file that contains the configuration information for the FPGA device. It configures the FPGA device, facilitates design verification and optimization, enables integration with the system, and provides flexibility in deployment and programmability. It is a critical process in realizing the desired functionality and performance of an FPGA-based system. We then select Export File to export the hardware information. It contains PS configuration information. Also, we must select to include the generated bitstream. Then we select Launch Software Development Kit (SDK). In SDK, we open a New Application Project with the Empty Application Feature, which enables us to add the source files we need. The Edge Zynq board appears as shown in Fig. 3.

Now, we connect the Edge Zynq board to the system, connect the board with the system that powers up the board, and then connect a monitor with an HDMI cable. With the board connected to the system, we can now program the board with our code to work for our functionality. To view our results, we run the project. This will start the camera, and the output of the image will be observed in the display.

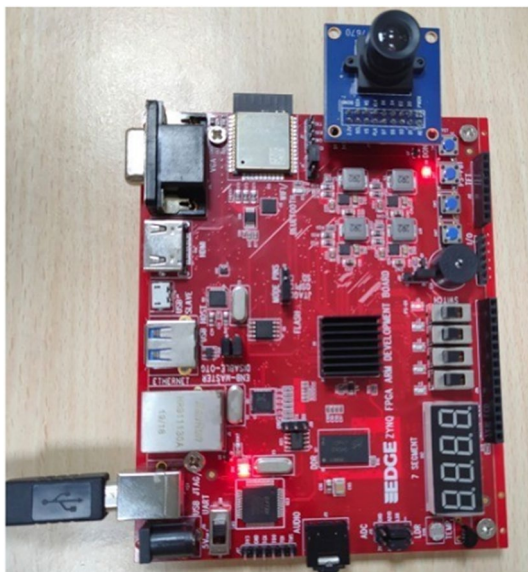


Fig. 3 Edge Zynq 7010 FPGA board.

## Results and Analysis

The synthesis and implementation process in the Vivado software environment begins with the creation of a bitstream file once the design and validation steps have been completed. This bitstream contains the configurable hardware design and is subsequently included in the project for export to the desired hardware platform. After exporting the bitstream to the device, we start the SDK program. The SDK offers a full development environment for customizing the target board to meet unique requirements. Here, we will develop and compile the software code essential to control physical components and perform specified tasks. After creating the SDK environment, the next step is to integrate the necessary source files into the project. These source files contain the code modules that interface with the OV7670 camera module, implement the edge detection method, and control the video output pipeline. After completing the project setup, we proceed to execute the project within the SDK environment. This includes compiling the programming for the whole project and initializing the hardware components. As a consequence, the camera module is engaged, recording live video feeds as shown in Figs. 4 and 5, which are later analyzed using the established edge detection algorithm. Finally, the processed video output is shown on the specified monitor giving a real-time preview of the collected video stream with edge detection enabled as shown in Fig. 6. This iterative procedure enables the smooth integration of hardware and software components, resulting in the effective implementation of our system's capabilities.

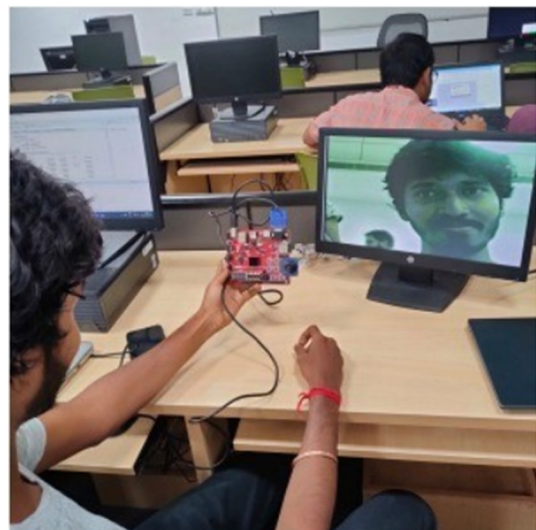
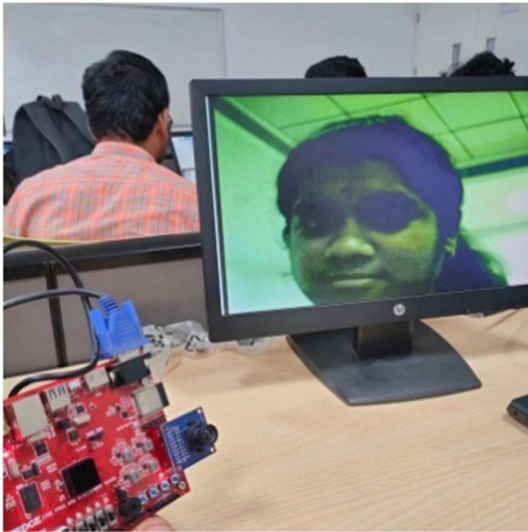
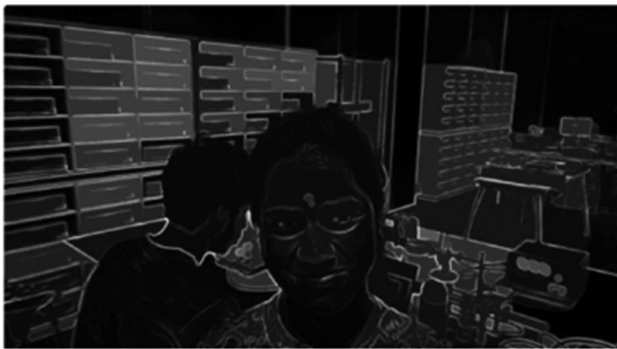


Fig. 4 Live image capturing.



**Fig. 5** Live image capturing.



**Fig. 6** Edge detection for captured images.

## Conclusion and Future Scope

The proposed technique represents a significant advancement in FPGA-based SoC methods, especially in the field of real-time edge detection. We successfully created and implemented a complex system capable of collecting live video feeds from the OV7670 camera module and transmitting them over HDMI, combining the computational capabilities of a Zynq processor and the flexible Vivado software package. This breakthrough is a critical step toward real-time video processing and transmission. In addition, by incorporating an edge detection algorithm into the system, we add another layer of capability that allows us to extract crucial information from collected photos. We solved frequent problems such as excessive edges in smooth areas and a lack of critical edges in detailed parts by implementing the Canny method at the block level. This upgrade ensures that the performance of our system's

edge detection capabilities remains high, without sacrificing accuracy or efficiency.

Looking ahead, we anticipate significant enhancements to our system. We want to incorporate edge detection directly into the live video stream, allowing for the detection of dynamic changes while emphasizing only essential features. This iterative development will increase the system's usability and applicability considerably, especially in situations where real-time analysis of changing settings is critical.

Furthermore, we anticipate our technology being used in a variety of applications beyond its existing capabilities. For example, in satellite imaging for remote sensing and observation, our system's capacity to recognize and highlight critical aspects in real time might be extremely useful for monitoring environmental changes, identifying possible threats, and supporting informed decision-making. In essence, our findings demonstrate the promise of FPGA-based SoC solutions for handling difficult computational tasks, notably real-time video processing and analysis. With continued improvement and refinement, our technology has the potential to revolutionize how we interact with and extract insights from visual data in a wide range of applications.

## Limitations

While our method is novel, it has several limitations that should be noted. The Zynq 7010 FPGA board we used has limited processing power. Complex edge detection techniques or high-resolution video feeds may exceed its real-time processing capabilities. Furthermore, our solution is dependent on particular hardware components, such as the OV7670 camera module and HDMI output, which may limit interoperability with different cameras or display interfaces. Setting and programming the FPGA board can be difficult and requires expertise. Despite gains in edge recognition accuracy at the block level using the Canny algorithm, difficulties with recognizing edges, particularly in noisy images, may continue. Delays in data transport and processing may also impede real-time performance. Finally, the capacity to manage many cameras or greater processing requirements may be limited without significant modifications. These factors are critical for determining the usefulness and breadth of our system.

**Acknowledgments** We acknowledge the support from the Department of Science and Technology-Science and Engineering Research Board (DST-SERB), Government of India ([CRG/2022/007866]), for funding the project.

**Conflict of interest** The authors declare that they have no conflict of interest.

## References

1. P. Ganesan and G. Sajiv, A comprehensive study of edge detection for image processing applications, in *2017 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS)* (2017). <https://doi.org/10.1109/ICIIECS.2017.8275968>, pp. 1–6.
2. A. Geetha Devi, B. Surya Prasada Rao, S. Abdul Rahaman, and V. Sri Sai Akhileswar, Real-time video image edge detection system, in *International Conference on Ubiquitous Computing and Intelligent Information Systems* (Springer, 2021), pp. 389–397.
3. X. Wei, G.M. Du, X. Wang, H. Cao, S. Hu, D. Zhang, and Z. Li, FPGA implementation of hardware accelerator for real-time video image edge detection, in *2021 IEEE 15th International Conference on Anti-counterfeiting, Security, and Identification (ASID)* (IEEE, 2021), pp. 16–20.
4. I. Nita, T. Costachioiu, V. Lazarescu, and T. Seceleanu, Multi-processor real-time edge detection using FPGA IP cores, in *2011 IEEE 7th International Conference on Intelligent Computer Communication and Processing* (IEEE, 2011), pp. 331–334.
5. N. Gajjar, V. Patel, and A.J. Shukla, Implementation of edge detection algorithms in real time on FPGA, in *2015 5th Nirma University International Conference on Engineering (NUICONE)* (IEEE, 2015), pp. 1–4.
6. P.K. Chakravathi, K.V. Kumar, P.D. Pradeep, and D. Suresh, FPGA based architecture for real-time edge detection. In *2015 Global Conference on Communication Technologies (GCCT)* (IEEE, 2015), pp. 12–17.
7. Z. Tan and J.S. Smith, Real-time canny edge detection on FPGAs using high-level synthesis, in *2020 7th International Conference on Information Science and Control Engineering (ICISCE)* (IEEE, 2020), pp. 1068–1071.
8. R. Jeyakumar, M. Prakash, S. Sivanantham, and K. Sivasankaran, FPGA implementation of edge detection using Canny algorithm, in *2015 Online International Conference on Green Engineering and Technologies (IC-GET)* (IEEE, 2015), pp. 1–4.
9. Q. Xu, S. Varadarajan, C. Chakrabarti, and L.J. Karam, A distributed canny edge detector: algorithm and FPGA implementation. *IEEE Trans. Image Process.* 23, 2944 (2014).
10. V. Raghavendra and L. Shrinivasan, Time efficient design and FPGA implementation of distributed canny edge detector algorithm, in *2018 3rd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT)*, 2135–2139 (IEEE, 2018).
11. N.M. Yusoff, I.S.A. Halim, N.E. Abdullah, et al., Real-time Hevea leaves disease identification using Sobel edge algorithm on FPGA: a preliminary study, in *2018 9th IEEE Control and System Graduate Research Colloquium (ICSGRC)* (IEEE, 2018), pp. 168–171.
12. J. Tian, J. Wu, and G. Wang, Realization of real-time Sobel adaptive threshold edge detection system based on FPGA, in *2015 IEEE International Conference on Information and Automation* (IEEE, 2015), pp. 2740–2743.
13. L. Guo and S. Wu, FPGA implementation of a real-time edge detection system based on an improved canny algorithm. *Appl. Sci.* 13, 870 (2023).

**Publisher's Note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.