R.V. COLLEGE OF ENGINEERING, BENGALURU – 560059
(Autonomous Institution Affiliated to VTU, Belagavi)

LOW POWER VLSI DESIGN

(Professional Core Elective-III)

(21EC64D3)

ASSIGNMENT REPORT

Edge Detection using Sobel Algorithm on FPGA

Submitted by,

Pavan Kumar C Banasode	1RV21EC116
Pratham G	1RV21EC128
Pradyumna S Athreya	1RV21EC119
Shreyas S	1RV21EC142

6th Sem B.E.

Electronics and Communication Engineering

2023-24

CONTENTS

ABSTRACT

This project presents a detailed implementation and analysis of an FPGA-based edge detection system using the Sobel algorithm. The design leverages Xilinx Vivado for hardware description and synthesis, coupled with Vivado SDK for software development. Implemented on a Zynq-7000 SoC, the system harnesses the power of programmable logic in conjunction with an ARM-based processor system. The edge detection algorithm, written in Verilog, is synthesized as a custom IP and integrated into a larger system design. Key operations include memory access, the convolution process involving two kernels, and the integration of these components into a top module. Additionally, a FIFO IP core was employed to optimize memory handling, crucial for high-speed data processing.

The design was rigorously tested and verified using a comprehensive testbench, and the RTL code was successfully exported as an IP core. This IP was then integrated into a block design utilizing the Zynq-7 Processing System and other necessary peripherals. Following the generation of the bitstream, the hardware configuration was exported, and software development was carried out in Vivado SDK to manage the PS and facilitate communication with a host PC. The project successfully demonstrated the FPGA's ability to perform real-time edge detection, with processed images transferred back to the PC via UART communication. The results underscore the efficacy of FPGAs in image processing applications, showcasing significant performance improvements over software-only implementations and paving the way for advanced real-time image processing.

INTRODUCTION

Edge detection is a fundamental technique in the field of image processing and computer vision, essential for identifying the boundaries and shapes of objects within an image. The Sobel operator is a popular method for edge detection, known for its simplicity and effectiveness in detecting edges in both horizontal and vertical directions

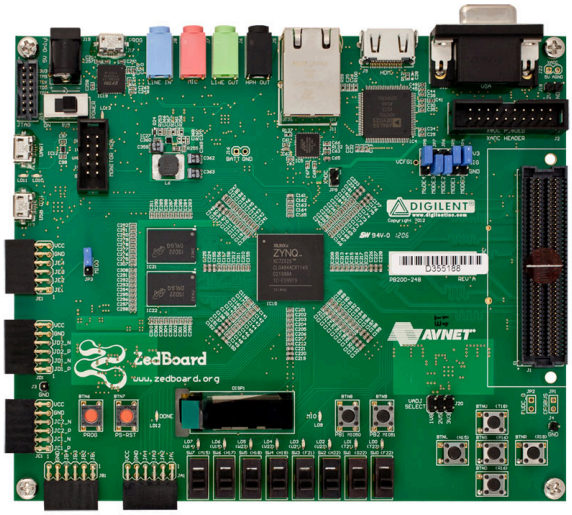

Fig. 1: ZedBoard

[1]. In the context of real-time systems, implementing such algorithms on general-purpose processors can lead to performance bottlenecks due to the extensive computational requirements [2]. FPGAs offer a compelling alternative due to their inherent parallel processing capabilities, allowing for the acceleration of image processing tasks.

This project focused on implementing the Sobel edge detection algorithm on an FPGA using the Zynq-7000 SoC, which combines FPGA fabric with a dual-core ARM Cortex-A9 processor. This architecture provides a versatile platform for implementing and testing hardware-accelerated algorithms while leveraging the processing power of the ARM cores for system management and peripheral control. The choice of the ZedBoard, a development board featuring the Zynq-7000 SoC, was driven by its rich feature set, including a flexible I/O system, high-performance FPGA fabric, and extensive support for embedded development [3,4]. The FPGA fabric was used to implement the Sobel algorithm, while the ARM cores handled tasks such as UART communication, image data

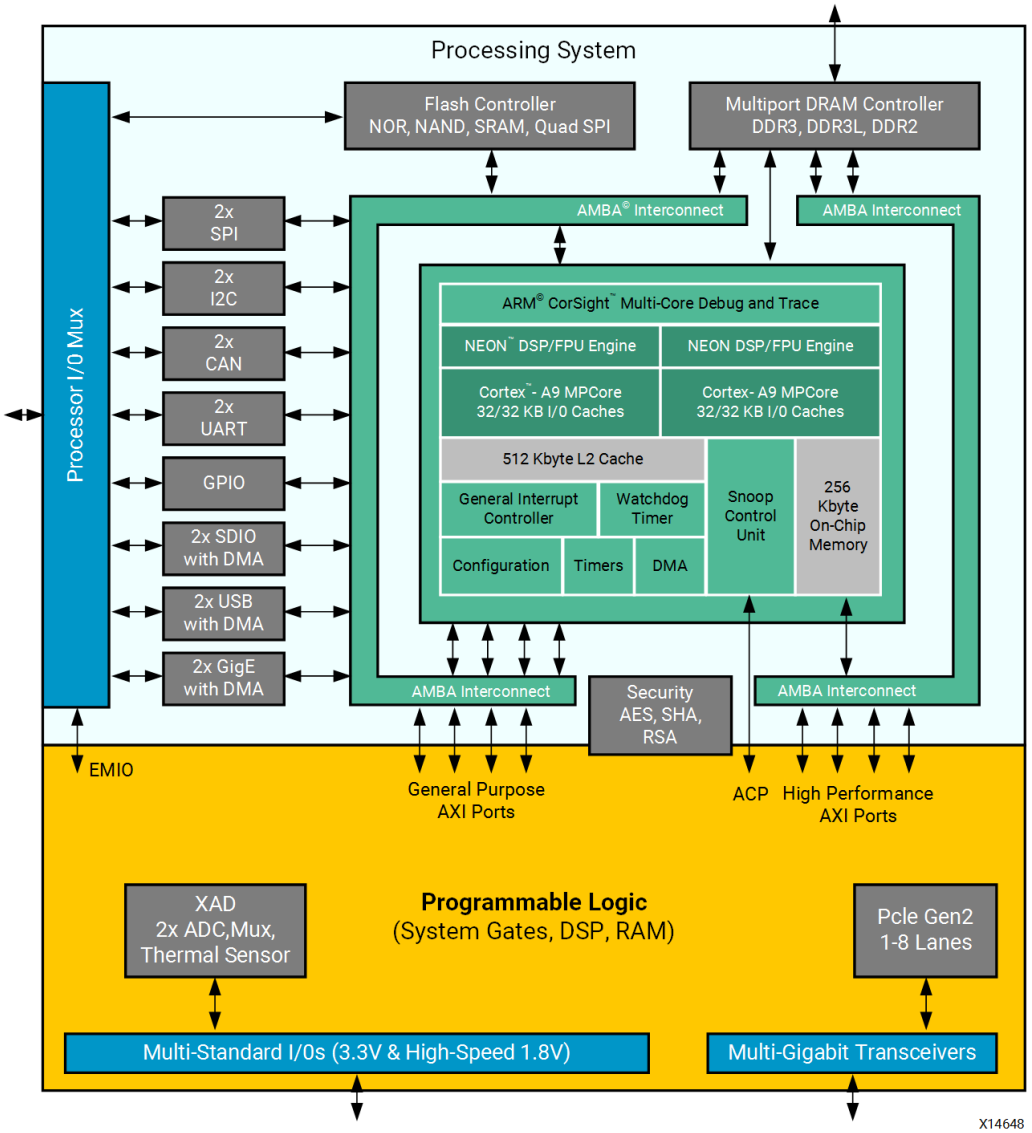

Fig. 2: ZedBoard System Architecture

transfer, and overall system control.

The significance of this project lies in its exploration of the FPGA's ability to handle computationally intensive image processing tasks in real-time. By implementing the Sobel edge detection algorithm on hardware, the project aimed to demonstrate the advantages of FPGAs in terms of speed, parallelism, and efficiency. The report provides a detailed account of the design process, from the initial development of the Verilog code to the final integration and testing of the system. The results obtained from the project highlight the effectiveness of FPGAs in real-time image processing, making them an ideal choice for applications where speed and accuracy are critical.

1. Gradient and Edge Detection

- Gradient: The gradient of an image measures the change in intensity (brightness) of the image. It is a vector with both magnitude and direction.
- Edge Detection: Edges in an image correspond to significant changes in intensity. By detecting these changes, edges can be identified.

2. Sobel Operator

The Sobel operator is a discrete differentiation operator, computing an approximation of the gradient of the image intensity function. It is implemented using two 3x3 convolution kernels, one for the horizontal direction (**Gx**) and one for the vertical direction (**Gy**).

$$G_x = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \quad G_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

$$G = \sqrt{Gx^2 + Gy^2}$$

These kernels are designed to respond maximally to edges running vertically and horizontally relative to the pixel grid, respectively.

LITERATURE SURVEY

FPGA-based edge detection has seen significant advancements over the past two decades. Neoh and Hazanchuk [5] pioneered real-time Sobel edge detection on FPGAs, demonstrating a system capable of processing video streams at 60 fps on a Xilinx Virtex-II. This early work established the feasibility of hardware-accelerated image processing for real-time applications.

Building on this foundation, Yasri et al. [6] proposed an optimized architecture for Sobel edge detection. Their implementation on a Xilinx Artix-7 FPGA achieved impressive performance, processing 1024x1024 images at 250 fps. This work highlighted the potential for high-performance edge detection even on mid-range FPGA devices, emphasizing efficient resource utilization and clever pipelining techniques.

The advent of heterogeneous computing platforms led to new approaches in FPGA-based image processing. Abirami et al. [7] explored a hardware-software co-design methodology using the Xilinx Zynq platform, which integrates an ARM processor with FPGA fabric. Their implementation of Sobel edge detection leveraged both hardware acceleration and software flexibility, achieving a 5x speedup compared to software-only solutions. This hybrid approach demonstrated the benefits of combining FPGA parallelism with the programmability of general-purpose processors.

METHODOLOGY

The methodology employed in this project encompasses both hardware and software design aspects, followed by system integration and testing. This comprehensive approach ensures a robust and efficient implementation of the edge detection system.

Hardware Design

The hardware design process began with the implementation of the Sobel edge detection algorithm in Verilog. This involved creating several key modules: a memory access module for storing and retrieving input image data, convolution modules for applying the horizontal and vertical Sobel kernels, and a top module to integrate these components [8,9]. Additionally, a FIFO IP core was utilized for efficient data buffering, ensuring smooth data flow through the processing pipeline.

To verify the functionality of the Verilog modules, a comprehensive testbench was developed. This testbench simulated the entire edge detection process, allowing for thorough validation of the design. The simulation results, showing both input and output images, provided crucial feedback for refining the implementation.

Once the Verilog implementation was verified, the next step involved creating a custom IP core from the RTL design. This custom IP encapsulated the entire edge detection functionality, making it easier to integrate into the larger system design.

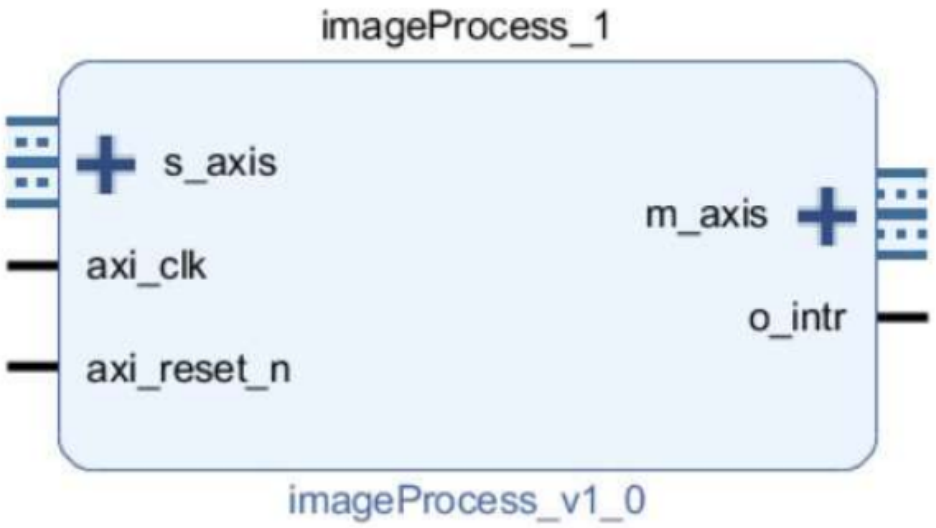

Fig. 3: Custom IP for Edge detection

The system integration phase utilized Vivado's block design feature to create a comprehensive system architecture. This design incorporated the custom edge detection IP, the Zynq Processing System, and other necessary peripherals such as UART and memory controllers. Careful attention was paid to the configuration of IP interconnections and clock domains to ensure proper system operation.

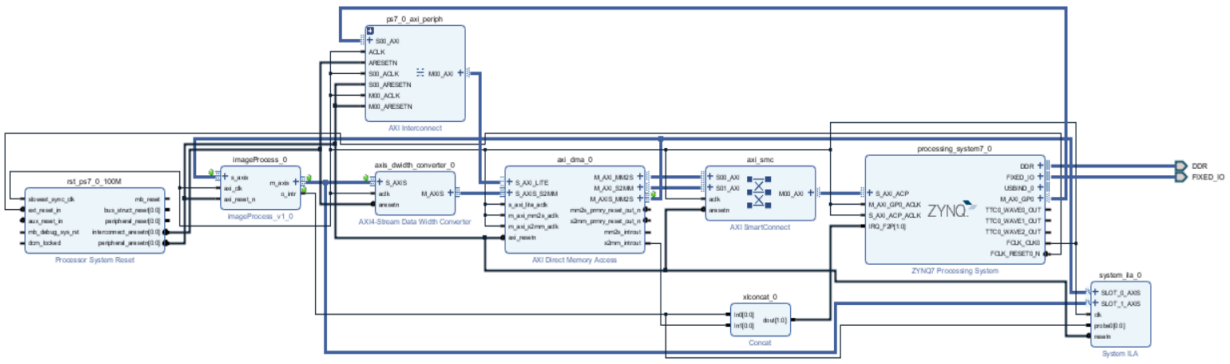

Fig. 4: Edge Detection Block Diagram

Key Components and Connections:

1. Processor System Reset:
    - o Handles reset signals for various system components.
    - o Ensures proper initialization and reset functionality.
2. Zynq Processing System:
    - o Central processing unit (CPU) of the Zynq-7000 SoC.

- Manages system control, configuration, and software execution.
- Interfaces with DDR memory for storing and retrieving data.

3. AXI Interconnect:
   - Connects various AXI masters and slaves within the system.
   - Facilitates data transfer between the processing system and custom IP block.
4. AXI Direct Memory Access (DMA):
   - Transfers data between memory and the custom IP block.
   - Uses AXI stream interfaces for efficient data movement.
5. AXI-Stream Data Width Converter:
   - Adjusts data width between different AXI stream components.
   - Ensures compatibility between components with varying data bus widths.
6. AXI SmartConnect:
   - Provides flexible and efficient interconnect solutions.
   - Supports multiple master and slave connections, optimizing data routing.
7. System ILA (Integrated Logic Analyzer):
   - Debugging tool for monitoring and analyzing AXI stream transactions.
   - Helps in verifying the functionality and performance of the custom IP block.

Data Flow:

1. Input Data:
   - Data is read from the DDR memory by the AXI DMA and sent to the custom IP block via the s_axis interface.
2. Image Processing:
   - The imageProcess_v1_0 block processes the input data (e.g., applying Sobel edge detection).
   - Processed data is output through the m_axis interface.
3. Output Data:
   - The processed data is transferred back to the DDR memory by the AXI DMA.
   - The CPU in the Zynq processing system can access the processed data for analysis
4. Interrupt Handling:
   - The o_intr signal from the custom IP block notifies the processor when processing is complete or if an event occurs, allowing for efficient interrupt-driven processing. This system design efficiently integrates custom image processing IP with other hardware and software components, leveraging the capabilities of the Zynq-7000 SoC for real-time image processing applications.

The final step in the hardware design process involved generating the bitstream for the complete design. This encompassed synthesis, implementation, and bitstream generation, resulting in a configuration file ready to be loaded onto the FPGA [10].

Software Design

In Vivado SDK, software was developed to control the Zynq-7 PS, managing tasks such as image data transfer and communication with the host PC. The software was responsible for sending the input image data to the FPGA, initiating the edge detection process, and receiving the processed image back from the FPGA via UART. This software development phase was crucial in enabling the system to function as a complete, integrated solution.

The PS control software was responsible for initializing both the PS and Programmable Logic (PL) components [11]. This included configuring the UART interface for communication with the PC, setting up DMA channels for efficient data transfer between the PS and PL, and implementing control logic to trigger the edge detection process and manage data flow.

The UART communication module was designed to handle the transmission and reception of image data between the FPGA and the host PC. This involved implementing a robust protocol for data transfer, including error handling mechanisms and data integrity checks to ensure reliable communication.

System Integration and Testing

The final phase of the project focused on integrating the hardware and software components and conducting comprehensive system testing. This began with programming the FPGA by loading the generated bitstream onto the Zedboard.

To facilitate communication between the PC and FPGA, PC-side software was developed to handle UART communication. This software implemented the necessary protocols for transmitting input images to the FPGA and receiving processed images back [12, 13].

The end-to-end testing process involved sending various test images from the PC to the FPGA, performing edge detection on the FPGA, and then receiving and displaying the processed images back on the PC. This thorough testing approach allowed for verification of the entire system pipeline, from data input to final output.

The final phase of the project involved testing the entire system on the ZedBoard. The image data was sent to the FPGA, where the Sobel edge detection algorithm was applied in real-time. The processed image was then transmitted back to the host PC, where it was displayed for analysis. The entire process was monitored to ensure that the system performed as expected, with the edge detection being accurate and the communication between the FPGA and PC being reliable.

RESULTS AND DISCUSSIONS

The results obtained from this project underscore the effectiveness of FPGAs in performing real-time image processing tasks. The Sobel edge detection algorithm was successfully implemented on the FPGA, achieving high-speed processing of image data with minimal latency. The output images, received back on the PC, displayed clear and

accurate edges corresponding to the original input images, demonstrating the algorithm's effectiveness in highlighting object boundaries.

One of the key advantages observed in this project was the FPGA's ability to process image data in parallel, significantly speeding up the edge detection process compared to traditional software-based approaches [14]. The use of a FIFO IP core for memory access played a crucial role in maintaining data flow and minimizing latency, ensuring that the system could handle large images without performance degradation. The Zynq-7 PS was also effective in managing the overall system, particularly in handling the communication between the FPGA and the host PC via UART [15].

Fig. 5: Bitmap images of before and after edge detection

However, the project also revealed some challenges associated with FPGA-based image processing. One of the primary challenges was ensuring efficient memory access, particularly when dealing with large images. While the FIFO IP core helped mitigate some of these challenges, further optimization could be explored to improve memory handling and reduce latency.

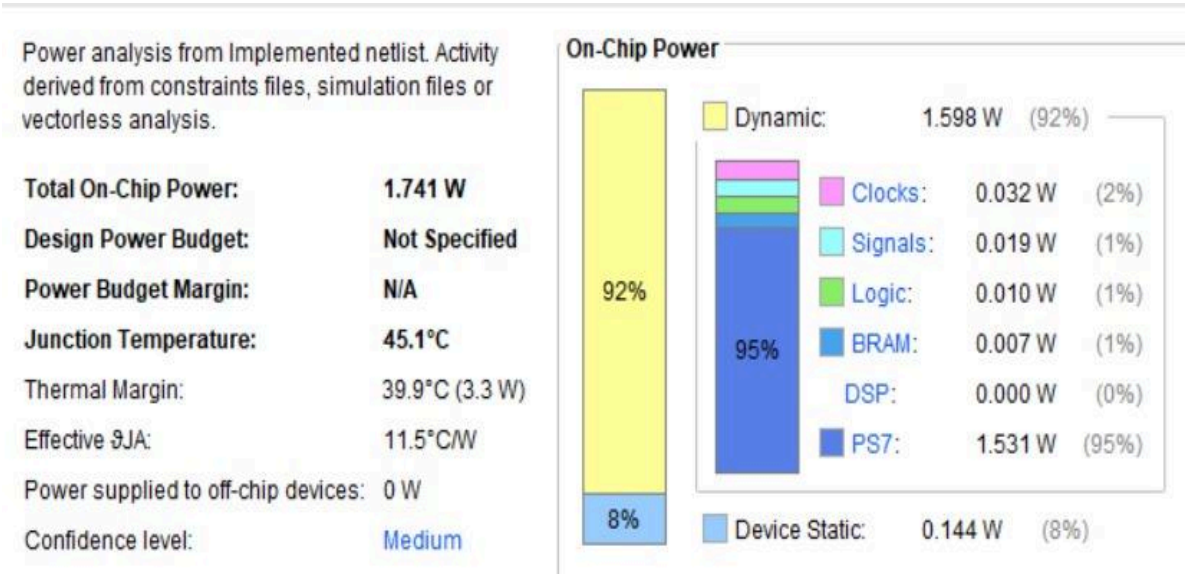

Fig. 6: Power Analysis from generated Netlist's Activity

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	0.531 ns	Worst Hold Slack (WHS):	0.028 ns	Worst Pulse Width Slack (WPWS):	3.750 ns
Total Negative Slack (TNS):	0.000 ns	Total Hold Slack (THS):	0.000 ns	Total Pulse Width Negative Slack (TPWS):	0.000 ns
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	32642	Total Number of Endpoints:	32642	Total Number of Endpoints:	10404

All user specified timing constraints are met.

Fig. 7: Timing analysis from generated Netlist's Activity

The success of this project opens up several avenues for future work. The Sobel edge detection algorithm could be extended to more complex image processing tasks, such as object recognition or real-time video processing. Additionally, the system could be optimized to handle higher-resolution images or to process multiple images simultaneously, further demonstrating the scalability of FPGA-based solutions. The modular nature of the design also allows for the integration of additional image processing algorithms, creating a versatile platform for real-time image analysis.

CONCLUSION

This report has presented the implementation of the Sobel edge detection algorithm on an FPGA using Vivado and Vivado SDK. The project successfully demonstrated the FPGA's capability to perform real-time image processing, leveraging its parallel processing power to achieve high-speed edge detection. The use of the Zynq-7000 SoC provided a flexible and powerful platform for the implementation, combining FPGA fabric with ARM cores to create an integrated solution for image processing.

The project also highlighted the importance of efficient memory handling and system integration in FPGA-based designs. The use of a FIFO IP core for memory access and the creation of a custom IP core for the Sobel algorithm were key factors in the system's success. The results obtained from the project demonstrated the accuracy and efficiency of the FPGA in performing edge detection, with the processed images being

Fig. 5: Bitmap images of before and after edge detection

reliably transferred back to the host PC via UART communication.

Looking forward, the project serves as a proof of concept for the use of FPGAs in real-time image processing applications. The modular design and scalability of the system open up opportunities for further development, including the implementation of more complex algorithms and the optimization of existing components. The experience gained from this project provides valuable insights into the challenges and opportunities of FPGA-based image processing, paving the way for future innovations in this field.

REFERENCES

[1] R. C. Gonzalez and R. E. Woods, Digital Image Processing, 4th ed. New York, NY: Pearson, 2018.

[2] S. Hauck and A. DeHon, Reconfigurable Computing: The Theory and Practice of FPGA-Based Computation. Burlington, MA: Morgan Kaufmann, 2008.

[3] L. H. Crockett, R. A. Elliot, M. A. Enderwitz, and R. W. Stewart, The Zynq Book: Embedded Processing with the Arm Cortex-A9 on the Xilinx Zynq-7000 All Programmable Soc. Glasgow, Scotland: Strathclyde Academic Media, 2014.

[4] K. Benkrid and D. Crookes, "From application descriptions to hardware in seconds: a logic-based approach to bridging the gap," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 12, no. 4, pp. 420-436, Apr. 2004.

[5] H. S. Neoh and A. Hazanchuk, "Adaptive edge detection for real-time video processing using FPGAs," Global Signal Processing, vol. 7, no. 3, pp. 2-3, 2004.

[6] I. Yasri, N. H. Hamid, and V. V. Yap, "An FPGA Implementation of Gradient Based Edge Detection Algorithm Design," International Journal of Computer and Electrical Engineering, vol. 6, no. 1, pp. 21-25, 2014.

[7] S. Abirami, C. Balasubramanian, and S. Rajesh, "Implement Edge Detection Using Xilinx System Generator," International Journal of Engineering Research and Technology, vol. 5, no. 4, pp. 332-335, 2016.

[8] M. S. Siddiqui, W. A. Serdijn, and J. Long, "A 1080p 60fps H.264/AVC video decoder with Canny edge detection implemented on Virtex-6 FPGA," in 2015 IEEE International Symposium on Circuits and Systems (ISCAS), Lisbon, 2015, pp. 1415-1418.

[9] C. Y. Lin, C. C. Cheng, C. H. Chao, K. H. Tsai, and S. J. Ruan, "An FPGA-based rapid whirlpool hash function," in Proc. 2009 Int. Conf. Parallel Process. Workshops, Vienna, Austria, 2009, pp. 83-88.

[10] K. Vipin and S. A. Fahmy, "FPGA dynamic and partial reconfiguration: A survey of architectures, methods, and applications," ACM Comput. Surv., vol. 51, no. 4, pp. 1-39, Jul. 2018.

[11] R. Ramanath, W. E. Snyder, and G. L. Bilbro, "Demosaicking methods for Bayer color arrays," J. Electron. Imaging, vol. 11, no. 3, pp. 306-315, Jul. 2002.

[12] J. Cong, B. Liu, S. Neuendorffer, J. Noguera, K. Vissers, and Z. Zhang, "High-level synthesis for FPGAs: From prototyping to deployment," IEEE Trans. Comput.-Aided Design Integr. Circuits Syst., vol. 30, no. 4, pp. 473-491, Apr. 2011.

[13] A. R. Brodtkorb, T. R. Hagen, and M. L. Sætra, "Graphics processing unit (GPU) programming strategies and trends in GPU computing," J. Parallel Distrib. Comput., vol. 73, no. 1, pp. 4-13, Jan. 2013.

[14] S. Jin, J. Cho, X. D. Pham, K. M. Lee, S. K. Park, M. Kim, and J. W. Jeon, "FPGA design and implementation of a real-time stereo vision system," IEEE Trans. Circuits Syst. Video Technol., vol. 20, no. 1, pp. 15-26, Jan. 2010.

[15] D. G. Bailey, Design for Embedded Image Processing on FPGAs. Singapore: John Wiley & Sons, 2011.