International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 08 Issue: 05 | May - 2024 SJIF Rating: 8.448 ISSN: 2582-3930

AN EFFICIENT VLSI IMPLEMENTATION OF EDGE DETECTION OF IMAGES

CH.JAYA PRAKASH[1]

G.GOKUL[2],M.MOULIDHAR[3],K.SRI NAVYA[4],K.SANDHYA[5],M.NAGA LAKSHMI[6]

Assistant Professor Sir C.R Reddy college of Engineering [1]

UG Scholars,Sir C R Reddy College of Engineering [2,3,4,5,6]

Abstact **– Edge detection is a fundamental process in image processing with significant implications across multiple domains. Within the realm of object detection, the accurate delineation of image perimeters forms the bedrock for subsequent analysis. In this study, we present an optimized Laplacian Edge Detection algorithm tailored for efficient implementation in hardware.Leveraging approximation methodologies and strategic pipelining, we streamline complex operations, thereby reducing latency and resource consumption. Our implementation, realized through the Verilog Hardware Description Language (VHDL), stands as a testament to the efficacy of our approach. Through rigorous optimization and meticulous comparison with conventional Sobel edge detection, we demonstrate superior performance metrics, including enhanced Area utilization, reduced Time Delay, and improved Power Efficiency. This work underscores the importance of methodological refinement and technological innovation in advancing edge detection techniques for practical applications.**

Keywords**:** Laplacian, Sobel, Kernel, Pixel range, Area, Power

I. INTRODUCTION

Edge detection, a cornerstone of image processing, is a pursuit deeply rooted in our human perception of visual cues and patterns. Much like how our eyes instinctively discern the outlines and boundaries of objects in our surroundings, edge detection algorithms aim to replicate this innate ability within the digital realm.

At its essence, edge detection embodies the essence of visual interpretation, seeking to identify the defining features that distinguish one element from another within an image. Just as we effortlessly recognize the edges of objects against contrasting backgrounds, these algorithms meticulously scrutinize pixel intensities, searching for abrupt changes that signify transitions between objects, textures, or surfaces. In the broad spectrum of image processing applications, edge detection holds unparalleled significance. It serves as the initial step in a myriad of tasks, from autonomous vehicle navigation to medical diagnosis and beyond. Much like how our brains rely on edges to parse complex scenes and make sense of our surroundings, edge detection algorithms provide the foundational framework for machines to interpret and understand visual information.Through this project, we embark on a journey to unravel the intricacies of edge detection, striving to bridge the gap between human perception and computational analysis. By delving into the theoretical underpinnings, exploring novel methodologies, and optimizing algorithms for hardware implementation, we aim to unlock new frontiers in visual intelligence.

Through this project, we embark on a journey to unravel the intricacies of edge detection, striving to bridge the gap between human perception and computational analysis. By delving into the theoretical underpinnings, exploring novel methodologies, and optimizing algorithms for hardware implementation, we aim to unlock new frontiers in visual intelligence.

Importance of Edge Detection-In the vast landscape of image processing, edge detection emerges as a beacon of

significance, guiding the way for numerous applications across various domains. Its importance transcends mere computational analysis, encapsulating the essence of human perception and understanding.

At its core, edge detection serves as the gateway to unlocking the wealth of information embedded within digital images. By isolating and delineating boundaries between objects, textures, and backgrounds, edge detection algorithms pave the way for a deeper understanding of visual data. Just as a painter outlines the contours of a subject to give it form and definition, these algorithms imbue digital images with structure and meaning. The implications of edge detection extend far beyond the realm of academic curiosity, permeating into countless real-world scenarios. In medical imaging, the ability to accurately detect edges enables clinicians to identify anomalies and diagnose diseases with unprecedented precision. In robotics and autonomous systems, edge detection facilitates navigation, object recognition, and scene understanding, empowering machines to interact with their environment intelligently.

Moreover, edge detection plays a crucial role in enhancing the efficiency and effectiveness of other image processing tasks. From image segmentation and feature extraction to pattern recognition and object tracking, the insights gleaned from edge detection serve as the building blocks for more complex analyses and decision-making processes. Through this project, we aim to unravel the intricacies of edge detection, shedding light on its theoretical foundations, practical applications, and potential for innovation. By harnessing the power of edge detection algorithms and optimizing them for hardware implementation, we endeavor to push the boundaries of what is possible in the realm of visual computing.

II. LITERATURE SURVEY

The integration of Laplacian edge detection alongside Sobel edge detection presents a nuanced examination of edge detection methodologies in image processing. While Sobel edge detection emphasizes gradient magnitude to identify edges, Laplacian edge detection focuses on detecting the zero-crossings in the second derivative of the image intensity[1]. This fundamental difference in approach leads to distinct characteristics in edge detection outcomes. Sobel operators, being gradient-based, are sensitive to noise and tend to produce thicker edges with a stronger emphasis on high-contrast transitions[2]. In contrast, Laplacian operators, being second-order derivative Laplacian operators, being second-order derivatives, offer advantages in detecting finer details and edges with smoother transitions, but they are more prone to detecting noise artifacts and require careful handling of thresholding parameters. Several studies have explored various approaches to integrate Laplacian and Sobel edge detection, including sequential application, weighted summation, and hierarchical combinations. Each approach has its trade-offs in terms of computational complexity and effectiveness in different image contexts. Further research is needed to develop more robust integration techniques and evaluate their performance across diverse datasets and application scenarios.

based, are more sensitive to rapid intensity changes and are adept at detecting finer details in edges.In terms of computational complexity, Sobel operators typically involve convolutions with two kernels to compute gradient magnitudes in both horizontal and vertical directions, followed by non-maximum suppression and thresholding[3]. On the other hand, Laplacian operators require convolutions with a single kernel representing the Laplacian of Gaussian (LoG) filter, which involves more intensive computations due to the convolution with the Gaussian kernel followed by the Laplacian operator[4]. However, this additional computational overhead often results in more precise edge localization and reduced sensitivity to noise compared to Sobel operators.Regarding implementation on hardware platforms such as FPGA, both Sobel and Laplacian edge detection algorithms can be optimized for efficient execution[5]. While Sobel operators lend themselves well to parallel processing and pipelining due to their simple convolutional operations, Laplacian operators may require more complex architectures to handle the convolution with the LoG filter efficiently. Additionally, the resource utilization and power efficiency of FPGA implementations may vary depending on the specific design choices and optimizations employed for each algorithm[6].In real-time applications, the choice between Sobel and Laplacian edge detection depends on the specific requirements of the task at hand. Sobel operators, with their simplicity and computational efficiency, may be preferred for tasks where real-time processing and low energy consumption are paramount, such as embedded vision systems in resource-constrained

environments. Conversely, Laplacian operators offer superior edge localization and noise robustness, making them suitable for applications where precise edge detection is critical, such as medical imaging or quality inspection in manufacturing[7].In conclusion, the comparison between Sobel and Laplacian edge detection algorithms underscores the importance of considering factors such as computational complexity, edge localization accuracy, and hardware implementation constraints when selecting the appropriate algorithm for a given application[8]. While Sobel operators offer simplicity and efficiency, Laplacian operators provide enhanced precision and robustness, catering to diverse requirements in image processing and computer vision tasks[9].

III. PROPOSED METHADOLOGY

Gradient-based edge detection methods utilize the computation of gradients or derivatives of pixel intensities to identify edges in an image. These methods are widely used due to their simplicity and effectiveness in detecting edges. Here are some common types of gradient-based edge detection methods
The Sobel operator is a popular gradient-based edge detection method that calculates the gradient magnitude for each pixel in an image. It uses a pair of 3x3 convolution kernels to compute gradients in the horizontal and vertical directions. The Sobel operator is particularly effective at detecting edges with high precision and is widely used in various image processing applications.

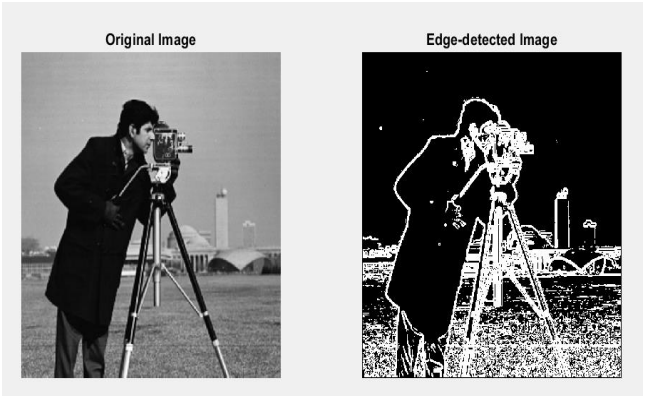
-1	**0**	**1**
-2	**0**	**2**
-1	**0**	**1**

Horizontal Sobel kernel

-1	**-2**	**-1**
0	**0**	**0**
1	**2**	**1**

Vertical Sobel kernel

The convolution of these kernels with the image results in two gradient images – one for the horizontal changes and the other for the vertical changes. The magnitude of the gradient at each pixel is then calculated to determine the overall edge strength. Sobel edge detection is widely used due to its simplicity and effectiveness in emphasizing edges in images. It serves as a fundamental step in various image processing applications, including object recognition, computer vision, and feature extraction.



Sobel Edge Detection Example

IV. OUT LINE

In the context of FPGA-based real-time image processing, the utilization of Laplacian-based edge detection systems presents a compelling approach to enhancing edge detection capabilities. Unlike Sobel operators, which rely on gradient-based methodologies, Laplacian operators offer a distinct advantage by focusing on second-order derivatives to pinpoint sharp transitions in pixel intensity. This method proves particularly advantageous in scenarios where noise poses a significant challenge, as Laplacian operators demonstrate greater resilience to noise interference compared to gradient-based approaches. By detecting zero-crossing points in the image, Laplacian edge detection algorithms excel in identifying subtle changes in brightness, facilitating more precise edge localization. Furthermore, the implementation of Laplacian operators requires relatively small hardware components, making them an efficient choice for FPGA-based systems. Through a VHDL-based architecture tailored for real-time edge detection, Laplacian-based systems offer a streamlined approach to identifying and delineating object boundaries, thereby contributing to advancements in fields reliant on accurate image analysis and recognition.

In addition to their noise resilience and precise edge localization capabilities, Laplacian-based edge detection systems offer simplicity in implementation, allowing for efficient utilization of FPGA resources. This simplicity facilitates rapid prototyping and deployment of real-time image processing solutions, making Laplacian operators an attractive choice for applications where speed and accuracy are paramount. Furthermore, the adaptability of Laplacian-based systems ensures their versatility across a wide range of image processing tasks, from object detection to scene segmentation. By leveraging the unique characteristics of Laplacian edge detection, FPGA-based systems can achieve enhanced performance and reliability, thus meeting the demanding requirements of modern image processing applications.

V. PROBLEMS IN EDGE ANALYSIS

In the realm of image analysis, the significance of object types in conveying meaningful information cannot be overstated. Images serve as repositories of somaticized data, where objects play a pivotal role in shaping the context and content of the visual narrative. Borders, viewed from a condensed perspective, manifest as discontinuities of magnitude, delineating the boundaries between distinct entities within the image. Natural images seldom exhibit sharp separations between adjacent regions; instead, they feature gradual transitions and subtle variations in pixel intensity. Graphic images, on the other hand, may employ synthesized phase edges to accentuate features and enhance visual clarity. Notably, slope edges exemplify the nuanced interplay between gradient changes and edge contours, showcasing the dynamic nature of natural edges influenced by factors like color and reflection in photography. Amidst this complexity, algorithms for edge detection play a pivotal role in discerning and delineating image borders. These techniques encompass a spectrum of approaches, ranging from linear filters to gradient-based methods, each tailored to address specific challenges and applications. By leveraging the diverse array of edge detection algorithms, image analysis systems can effectively extract pertinent information, enabling a deeper understanding and interpretation of visual data across various domains.

VI. IMPLEMENTATION OF THE PROJECT

Laplacian edge detection is a popular technique used in image processing to identify edges by computing the second derivative of pixel intensities. Unlike gradient-based methods, which detect edges by analyzing changes in intensity gradients, Laplacian edge detection focuses on detecting points of maximum intensity change within an image. In Laplacian edge detection, the Laplacian operator is applied to the image, which highlights regions where the intensity changes rapidly. This operator computes the sum of the second derivatives of pixel intensities in both the horizontal and vertical directions. By convolving the image with the Laplacian kernel, areas of rapid intensity change, corresponding to edges, are enhanced.

However, direct application of the Laplacian operator can amplify noise and produce spurious edge detections. Therefore, Gaussian smoothing is often applied to the image before applying the Laplacian operator. This helps to reduce noise and results in more accurate edge detection. One common approach to Laplacian edge detection is the Laplacian of Gaussian (LoG) method, which combines Gaussian smoothing with the Laplacian operator. The LoG operator provides better localization of edges and is less sensitive to noise compared to using the Laplacian operator alone.

Overall, Laplacian edge detection is widely used due to its simplicity and effectiveness in detecting edges in various images. However, it may struggle with noise and produce false detections in regions with low contrast or complex textures. Therefore, careful parameter tuning and preprocessing are often required to achieve optimal results in practical applications.

Laplacian-Based Edge Detection Techniques

Laplacian edge detection is a popular method used in image processing to identify edges by analyzing the second derivative of pixel intensities. Unlike gradient-based methods, which compute the gradient magnitude to detect edges, Laplacian edge detection focuses on detecting zero crossings in the second derivative of intensity values.

The Laplacian operator, also known as the Laplacian filter, is applied to the image to highlight areas of rapid intensity change, indicating the presence of edges. The operator is often represented as a 3x3 or 5x5 convolution kernel, which is convolved with the image to calculate the Laplacian response.One of the advantages of Laplacian edge detection is its ability to detect edges regardless of their orientation. It is particularly effective at detecting edges in regions with uniform intensity, where gradient-based methods may struggle. However, Laplacian edge detection is sensitive to noise, which can produce false edge detections. To address this issue, noise reduction techniques such as Gaussian smoothing are often applied before applying the Laplacian operator.In discrete form, the Laplacian can be approximated using convolution with a kernel. A common 3x3 Laplacian kernel is:

0	**1**	**0**
1	**-4**	**1**
0	**1**	**0**

Laplacian Kernel

The Laplacian operator is sensitive to rapid intensity changes, and it responds to edges regardless of their orientation. However, it also amplifies noise, so it is often used in combination with smoothing techniques, such as applying a Gaussian filter to the image before applying the Laplacian. The Laplacian edge detection process involves the following steps:

Laplacian Filtering: Convolve the image with the Laplacian kernel to compute the second derivative.

Edge Thresholding: Set a threshold to identify pixels with significant intensity changes as edges. Pixels with values above the threshold are considered part of the edges.

Edge Enhancement (Optional): Enhance the detected edges if necessary using techniques such as histogram equalization or other post-processing methods.

The Laplacian edge detection method is widely used in computer vision, image processing, and feature extraction applications. It provides a quick way to identify areas of significant intensity change in an image, which often correspond to object boundaries or edges.

Laplacian Edge Detected Image

VII. WORK FLOW

The below flowchart represents the workflow that is involved in making the image to a text file consisting of Pixel values.

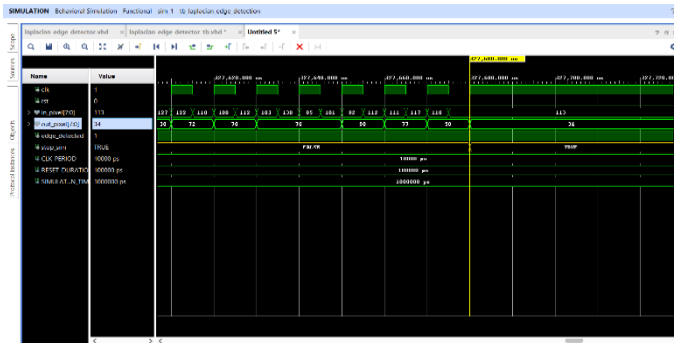
Load the image file "cameraman.jpg" into MATLAB using `imread`.Reshape the image matrix into a column vector and define the file path for saving pixel intensities. Write the pixel intensities to a text file and close the file using `fprintf` and `fclose`.Implement error handling using a `try-catch` block to manage potential file operation errors and display appropriate messages.

Vivado Workflow Algorithm

The binary image data is provided as a text file containing pixel intensity values.Define input ports in the VHDL module to receive pixel intensity values.Implement logic to read pixel values from the text file and buffer them for processing.Define a 3x3 array or buffer to store the pixel intensity values of the neighborhood window.

Update the window with new pixel values for each pixel iteration.Apply the Laplacian operator to the pixel neighborhood window to compute the Laplacian value for each pixel.

Implement the Laplacian operator as a convolution kernel



to calculate the second derivatives of pixel intensities.Calculate the magnitude of the Laplacian value to determine the presence of edges.Apply a threshold to classify pixels as edge or non-edge pixels.Mark pixels as part of an edge if the magnitude exceeds the threshold.Define output ports to transmit the computed magnitude of the Laplacian value for each pixel.

Assign the magnitude values to the output pixels to generate the edge-detected image.Implement clock and reset signals to synchronize the operations and initialize internal signals and registers.Use clock edge detection to trigger the processing of each pixel.Simulate the VHDL design using Vivado Simulator to verify the functionality of the Laplacian edge detection algorithm.Verify that the algorithm correctly detects edges in the input image and produces the expected output.

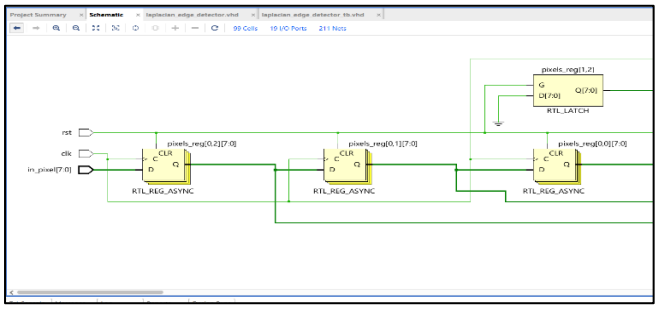
Run synthesis and implementation processes in Vivado to generate the hardware description files (bitstream) for the target FPGA device.Ensure that requirements.Compare the performance and accuracy of the implemented Laplacian edge detection algorithm with other edge detection methods.Evaluate factors such as edge detection quality, resource utilization, and computational efficiency.Apply optimization techniques to improve the efficiency and resource utilization of the VHDL implementation.

VIII. RESULTS

Simulation results of comparison after all pixels are completed.

The above figure of the simulation represents the stop_sim is kept to be true after all pixels are completed and the simulation stopped from there since there is no clock generated after it is kept as active.

Schematic RTL diagram:

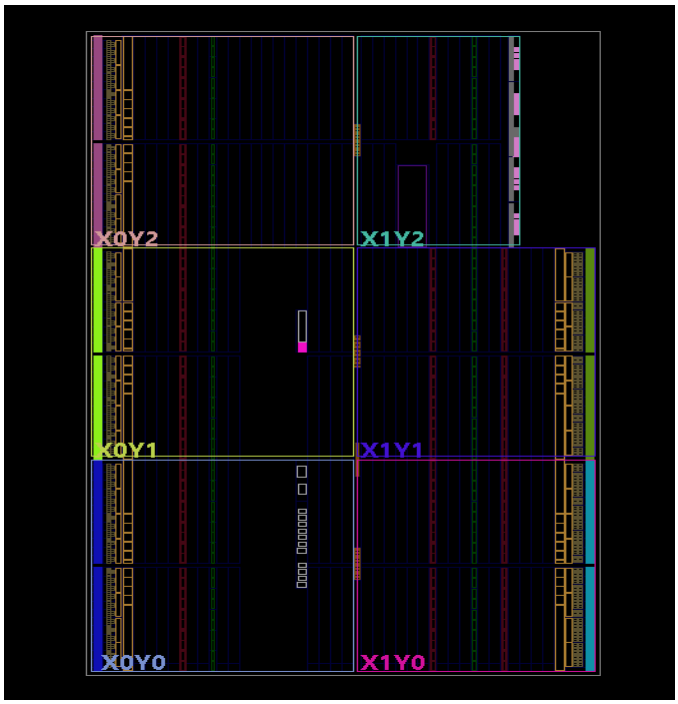


Input Block of schematics

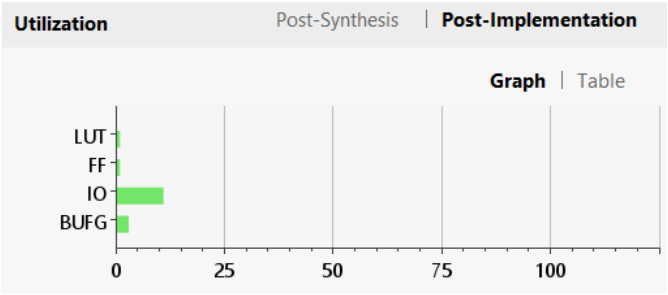
The provided figure depicts schematic block diagrams illustrating the system's architecture. In the second image, a magnified view of the input block reveals components such as the reset signal (rst), clock signal (clk), and an 8-bit input signal (in_pixel [7:0]).

Synthesis output:

The provided figures depict the synthesis design and corresponding schematics for the Laplacian edge detection algorithm. The first figure showcases the synthesis design, which represents the logical and physical implementation of the Laplacian edge detection algorithm within the Vivado environment. This design encapsulates the hardware description of the algorithm, detailing how it will be realized on the FPGA. In the



second figure, the schematic representation offers a visual



breakdown of the synthesized design. It provides insights into the internal components and connections within the

Resource	Utilization	Available	Utilization %
LUT	27	20800	0.13
FF	50	41600	0.12
IO	19	170	11.18
BUFG	1	32	3.13

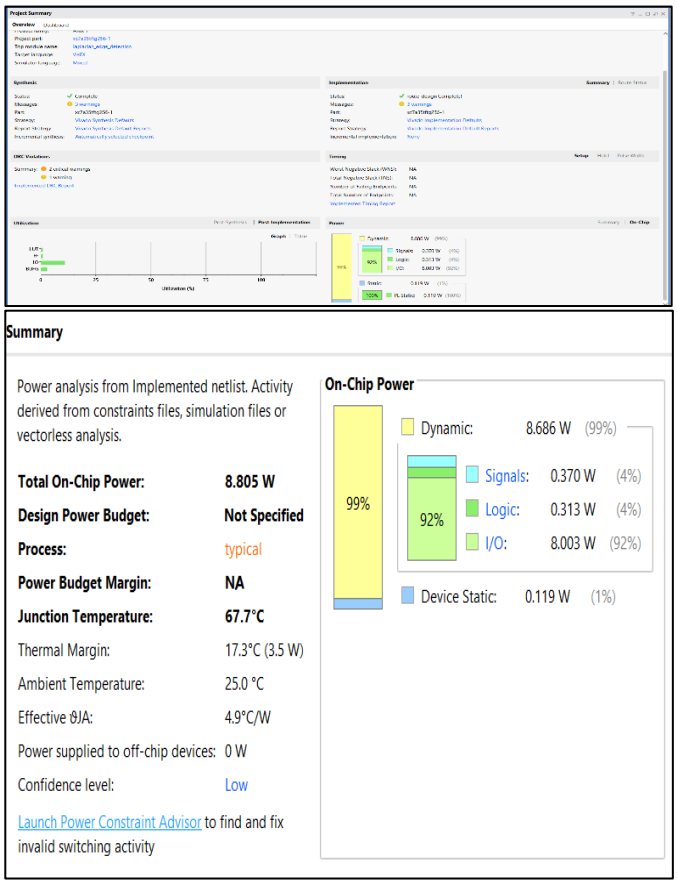
Laplacian edge detection module.

Project Summary:

The power consumption and area utilization of the design are computed through the implementation process, providing detailed representations of these metrics.

Additionally, the time delay associated with the design is also evaluated as part of this comprehensive analysis. This integrated approach enables a thorough assessment of the design's performance characteristics, ensuring its efficiency and effectiveness in real-world applications Through the implementation process, Vivado generates comprehensive reports and metrics, offering insights into the power consumption, area utilization, and time delay of the design. These reports provide detailed representations of each metric, allowing engineers to assess the design's efficiency and identify potential areas for optimization.

Power Report



The total on-chip power consumption is 8.805 W, derived from activity constraints, simulation, or vectorless analysis. The design operates within a power budget that is not specified, with a typical process. The junction temperature is 67.7°C, providing a thermal margin of 17.3°C. The ambient temperature is 25.0°C, resulting in an effective thermal resistance of 4.9°C/W. Power supplied to off-chip devices is 0 W. The confidence level is low, prompting a review of switching activity constraints using the Power Constraint Advisor to ensure accuracy and compliance with design requirements.

IX. AREA UTILIZATION

X.

Specifically, it reveals that:

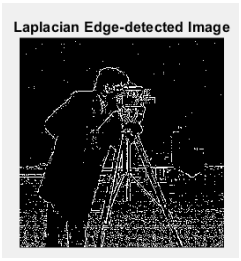
- LUT (Look-Up Table) utilization is 27 out of 20800, accounting for 0.13% of available LUTs.
- FF (Flip-Flop) utilization is 50 out of 41600, representing 0.12% of available flip-flops.
- IO (Input/Output) utilization is 19 out of 170, constituting 11.18% of available I/O resources.

- BUFG (Clock Buffer) utilization is 1 out of 32, occupying 3.13% of available clock buffers.

EDGE DETECTED IMAGE:

Original Image

Sobel Edge Detected Image



Laplacian Edge Detected Image

Power Comparison of Existing Sobel Model and proposed Laplacian Model:

XI. COMPARISIONS

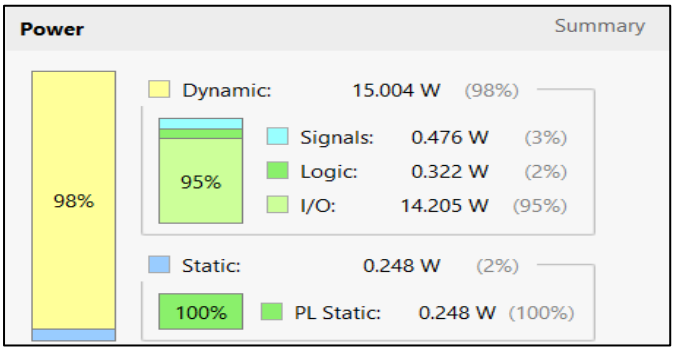
Power Constraint	Sobel Algorithm	Laplacian Algorithm
Total on-chip power	15.252 W	8.805W
Junction Temperature	99.0 ^{0} C	67.7 ^{0} C
Thermal Margin	-14^{0}C (-2.8 W)	17.3 °C (3.5 W)
Effective Thermal Resistance	4.9 ^{0}C/W0	4.9 °C/W

Power Comparison with power constraints

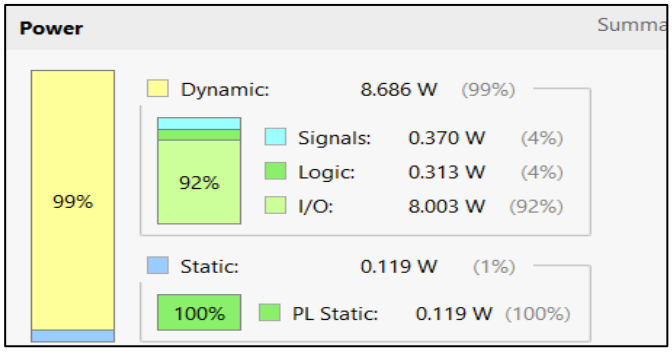
The comparison reveals that the Laplacian algorithm consumes significantly less total on-chip power compared to the Sobel algorithm, with a reduction from 15.252 W to 8.805 W. This reduction in power consumption correlates with a lower junction temperature for the Laplacian algorithm, indicating improved thermal performance. The thermal margin, which represents the difference between the junction temperature and the maximum allowable temperature, is notably higher for the Laplacian algorithm, indicating a larger margin of safety against overheating compared to the Sobel algorithm. Additionally, both algorithms exhibit the same effective thermal resistance, suggesting similar thermal conductivity properties despite differences in power consumption and temperature. Overall, the Laplacian algorithm demonstrates superior power efficiency and thermal performance compared to the Sobel algorithm, making it a favorable choice for power-constrained environments

Time Comparision

Sobel Detection	Laplacian Detection
6.241ns	5.289ns



Power Report of Sobel



Power Report of Laplacian

Power Comparison for the utilization resources

Type	Power used (Sobel)	Power used(Laplacian)
Signals	0.476 W	0.370 W
Logic	0.322 W	0.313 W
I/O	14.205 W	8.003 W
PL Static	0.248 W	0.119 W

Area Comparison of Existing Sobel Model and Proposed Laplacian Model

Resource	Available	Utilization by Sobel	Utilization by Laplacian
LUT	**20800**	**27**	**27**
FF	**41600**	**58**	**50**
IO	**170**	**28**	**19**
BUFG	**32**	**1**	**1**

XII. CONCLUSION

In summary, our project has meticulously crafted and optimized a Laplacian Edge Detection algorithm tailored for efficient hardware implementation. By employing sophisticated approximation methodologies and strategic pipelining techniques, we have successfully streamlined complex operations, significantly reducing latency and resource consumption. Implemented using the Verilog Hardware Description Language (VHDL), our solution exemplifies the fusion of computational accuracy and hardware efficiency crucial for edge detection in object recognition.

Throughout our research, we underscored the pivotal role of edge detection across various domains, particularly in object detection, where precise delineation of image perimeters underpins subsequent analysis and decision-making processes. By enhancing the efficiency and accuracy of edge detection algorithms, we propel advancements in fields like computer vision, robotics, autonomous systems, and medical imaging. A cornerstone of our contribution lies in optimizing the Laplacian Edge Detection algorithm for hardware implementation. Leveraging approximation methodologies, we achieved a delicate balance between computational precision and hardware efficiency. This optimization not only facilitates real-time processing but also reduces resource requirements, rendering it suitable for embedded systems and resource-constrained environments.

Moreover, our strategic pipelining approach maximizes hardware utilization and throughput while minimizing latency. By breaking down computations into manageable stages and overlapping their execution, we exploit inherent parallelism, enhancing performance metrics such as throughput rates and response times. These are critical for applications requiring rapid decision-making and continuous processing of image data streams.

In conclusion, our optimized Laplacian Edge Detection algorithm marks a significant stride towards efficient hardware implementations for object recognition tasks. By addressing challenges of computational intensity and resource constraints, we pave the way for integrating edge detection into diverse embedded systems, fostering technological advancements and societal progress.

XIII. REFERENCES

[1] Charu, Pardeep Kumar, Kuldeep Singh (2020) Hardware Model for Efficient Edge Detection in Images, 2020 IEEE International Conference on Computing, Power and Communication Technologies (GUCON) Galgotias University, Greater Noida, UP, India. Oct 2-4, 2020

[2] Siva Sankari B, Dr Viji.(2021) A. AN EFFICIENT VLSI IMPLEMENTATION OF EDGE DETECTION ALGORITHM FOR IMAGE PROCESSING APPLICATION, International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056

[3] [3] Kishor Kumar, Dr. Vijaya Lakshmi. D. (2022) An Efficient Implementation Of Edge DetectionAlgorithm For Image Processing Using Fpga. Journal of Positive School Psychology 2022, Vol. 6, No. 10, 3110-3119

[4] T. P. Lavanya, S. Premkumar (2014) An Efficient Approach for Edge Detection Hardware Accelerator for Real Time Video Segmentation using Laplacian Operator, International Journal of Engineering Research & Technology (IJERT) IJERT ISSN: 2278-0181 Vol. 3 Issue 11, November-2014

[5] T. Jagadesh (2020), Implementation of Prewitt Operator based Edge Detection Algorithm using FPGA. International Journal for Research in Applied Science & Engineering Technology (IJRASET) ISSN: 2321-9653

[6]] Abidha Abdul Karim Rawther, Leena Mary (2020). The Design of Low Power Sobel Edge Detection in FPGA. AIP Conference Proceedings 2222, 030010 (2020)

[7] Talal Bonny, and Safaa Henno(2018). Image Edge Detectors under Di®erent Noise Levels with FPGA Implementations. Journal of Circuits, Systems, and Computers Vol. 27, No. 13 (2018) 1850209.

[8] Yangli ,Yangbing. ―Study of FPGA based Parallel Processing of Sobel Operator‖ AI Modern Electronics Technique 2005.J. I.Yasri*, N.H.Hamid, V.V.Yap, 2008. "Performance Analysis of FPGA Based Sobel Edge Detection Operator" IEEE.

[9] Mohamed Nasir Bin Mohamed Shukor, Lo HaiHiung, Patrick Sebastian3, 2007. "Implementation of Real-time Simple Edge Detection on FPGA" pp. 1404-1405, IEEE.

[10] Heath M., Sarker S., Sanocki T. and Bowyer K.," Comparison of Edge Detectors: A Methodology and Initial Study", Proceedings of CVPR'96 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp.143-148, 1996.

[11] G. Anusha, T. JayaChandra Prasad, D. Satya Narayana., "Implementation of SOBEL Edge Detection on FPGA" International Journal of Computer Trends and Technology- volume (3) issue (3)- 2012

[12] R.Gonzalaz, R Woods, "Digital ImageProcessing", New Jersey: Prentice –Hall 2002.

[13] Chaple GN, Daruwala RD, Gofane MS. Comparisons of Robert, Prewitt, Sobel operator based edge detection methods for real time uses on FPGA. Paer presented at: 2015 International Conference on Technologies for Sustainable Development (ICTSD); 2015; Mumbai:1-4.

[14] Chaple G, Daruwala RD. Design of Sobel operator based image edge detection algorithm on FPGA. Paper presented at: 2014 International Conference on Communication and Signal Processing; 2014; Melmaruvathur:788-792.

[15] Israni S, Jain S. Edge detection of license plate using Sobel operator. Paper presented at: 2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT); 2016; Chennai:3561-3563.

[16] Amara AB, Pissaloux E, Atri M. Sobel edge detection system design and integration on an FPGA based HD video streaming architecture. Paper presented at: 2016 11th International Design & Test Symposium (IDT); 2016; Hammamet:160-164.

[17] Eetha S, Agrawal S, Neelam S. Zynq FPGA based system design for video surveillance with Sobel Edge Detection. Paper presented at: 2018 IEEE International Symposium on Smart Electronic Systems (iSES) (Formerly iNiS); 2018; Hyderabad, India:76-79.

[18] Xilinx: Vivado design suite: high-level synthesis; 2018. https://www.xilinx.com/support/documentation/ sw_ manuals/xilinx2018_3/ug902-vivado-high-level-synthesis.pdf. Accessed September 12, 2019.

[19] MathWorks HDL coder. https://www.mathworks.com/products/hdl-coder.html. Accessed August 14, 2019.

[20] Cong J, Liu B, Neuendorffer S, Noguera J, Vissers K, Zhang Z. High-level synthesis for FPGAs: from prototyping to deployment. *IEEE T Comput Aid D*. 2011;30:473-491.

[21] Choe S. ESL enables software driven SoCs. Coware Inc. https://www.design-reuse.com/articles/10952/eslenables-software-driven-socs.html. Accessed April 5, 2020.

[22] Bergamaschi RA, O'Connor RA, Stok L, et al. High-level synthesis in an industrial environment. *IBM J Res Develop*. 1995;39:131-148.

[23] Kucukcakar K, Chen CT, Gong J, Philipsen W, Tkacik TE. Matisse: an architectural design tool for commodity ICs. *IEEE Des Test Comput*. 1998;15:22-33.

[24] Lippens PER, van Meerbergen JL, van der Werf A, et al. PHIDEO: a silicon compiler for high speed algorithms. *ProceedingsoftheEuropeanConferenceonDesign Automation*. Amsterdam, The Netherlands: IEEE Computer Society Press; 1991:436-441.

[25] Biesenack J, Koster M, Langmaier A, et al. The Siemens high-level synthesis system CALLAS. *IEEE Trans Very Large Scale Integr Syst*. 1993;1:244-253.

[26] Knapp DW. *Behavioral Synthesis: Digital System Design Using the Synopsys Behavioral Compiler*. Englewood Cliffs, NJ: Prentice-Hall; 1996.

[27] https://www.mentor.com/hls-lp/catapult-high-level-synthesis. Accessed August 14, 2019.

[28] https://www.cadence.com/content/cadence-www/global/en_US/home/tools/digital-design-and-signoff/ synthesis/stratus-high-level-synthesis.html. Accessed August 14, 2019.

[29] MathWorks Sobel edge detection with vision HDL toolbox. https://www.mathworks.com/help/hdlverifier/ examples/sobel-edge-detection-algorithm-with-computer-vision-system-toolbox.html. Accessed April 5, 2020.

[30] Koyuncu I, Çetin Ö, Katırcıo˙ glu F, Tuna M. Edge detection application with FPGA based Sobel operator.˘ Paper presented at: 2015 23nd Signal Processing and Communications Applications Conference (SIU); 2015; Malatya:1829-1832.

[31] Singh S, Saini AK, Saini R, Mandal AS, Shekhar C, Vohra A. Area optimized FPGA-based implementation of the Sobel compass edge detector. *ISRN Mach Vis*. 2013;2013:1-6. https://doi.org/10.1155/2013/820216.