Design and Implementation of Edge Detection Algorithm for Image Processing

Pooja Kumari[1], Dr. Vijaya Prakash A M[2]
[1]M.Tech, (VLSI Design & Embedded System) BIT, Bangalore
[2]Professor, BIT, Bangalore

Abstract—**Digital image processing have become the mostly explored areas of modern technology due to its vast applications in medical imaging, surveillance, robotics, and computer vision. Traditionally, in many of the image edge detector algorithms are executed using techniques like computer simulation, which make testing and validation relatively simple. However, when algorithms or architectures grow in complexity, increases the time of simulation drastically. This makes the computational cost high and also makes software-based implementations unsuitable for real-time, high-speed image and video applications. To overcome this limitations, hardware-based solutions, particularly FPGA architectures, are adopted because of parallel processing capability, it is reconfigurable, and reduced latency. This work presents an FPGA-based realization of the Canny edge detection algorithm, incorporating adaptive thresholding to improve the performance of image processing applications. Unlike fixed thresholds, adaptive thresholding dynamically adjusts to varying illumination and noise conditions, thereby improving edge localization accuracy and robustness. The proposed architecture integrates Gaussian smoothing, gradient computation, non-maximum suppression, and adaptive thresholding, followed by edge tracking. The complete design is modeled and programmed in Verilog, simulated and synthesized using Xilinx Vivado tools. Experimental results demonstrate that the modified design reduces memory requirements and latency while maintaining high throughput and strong edge detection performance, outperforming conventional fixed-threshold Canny implementations.**

***Keywords*— Image Processing, HDL, MATLAB, Thresholding.**

I. INTRODUCTION

Detection of edges in image are significantly used in many of the domains, it includes computer vision, geological exploration, remote sensing through satellite, medical fields, aircraft, and transportation. Edge detection is characterized as a collection of mathematical techniques used to identify the boundaries within an image. At the edges, pixel intensity values change abruptly, creating noticeable discontinuities. These edges represent the outlines of objects of the image, which are beneficial for recognizing and classifying them. Ideally, applying an edge detector produces a set of connected edges that form the outline of objects, capturing their structure. Edge detection identifies edges in multiple orientations and is mostly used for significant features detection or events in image sequences. The occurrence of edges depends on factors such as irregularities in surface depth, changes in orientation, variations in material properties, and differences in illumination. Edge detection predominantly can be categorized into gradient-based and Laplacian-based approaches. The gradient-based method, also called the first-order derivative approach, computes gradients by differentiating the image. In contrast, the Laplacian method uses the second-order derivative for edge detection. Numerous researchers have created different methods for detecting edges. Robert, Prewitt, Sobel, Laplace of Gaussian and Canny edge detector are examples of edge detection techniques. The Roberts operator fails to capture finer edges, resulting in incomplete edge maps. Among various edge detection techniques, the Canny algorithm is considered most effective and has long been regarded as a benchmark method. The conventional implementations of the Canny edge detection technique, however employ the same fixed threshold i.e. high threshold and low threshold values for every input image, and the procedure is complicated. There are two approaches that can be used for obtaining threshold in hardware implementations: fixed threshold and adaptive threshold. One crucial problem with the Canny mask for edge detection is figuring out the right threshold with the least amount of latency and computing expense. Although using a fixed threshold helps simplify the process, performance suffers as a result. With the aim of achieving low latency and high throughput, a robust threshold computation approach, i.e the Canny edge detector using adaptive threshold is employed. By calculating low and high thresholds, it helps to maintain important edges in the high-detailed region while suppressing excessive edges in the smooth zone. The most crucial stage in edge detection is determining the object's edges and the image's pixel information. Two different kinds of masks one oriented horizontally and the other vertically will be used in each edge detection procedure. The convolution vector of the nxn matrix, which is multiplied by the image's sub-window, is the kernel or mask. These compute the rate of change of image intensity function. Edges correspond to locations where the gradient magnitude is high.

A. Edge detection Operator

1) Robert Cross operator:

$$\begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$$

2) Prewitt Operator:

$$\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}$$

3) Sobel Operator :

$$\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

A. Canny Edge Detection Algorithm

The approach for edge identification technique is widely utilized in processing of digital images due to its effectiveness. It follows the steps:

1. Input Pixel: The RGB image is first converted into a grayscale format, and a coefficient file is generated using MATLAB.
2. Smoothing: The grayscale image is then filtered with a Gaussian filter to minimize noise and ensure that random variations are not misinterpreted as edges.
3. Gradient Calculation: The gradient magnitude and orientation are computed for each pixel position.
4. Thresholding: Edge pixel detection is facilitated by utilizing a threshold generated from the gradient magnitude of the image.
5. Hysteresis Thresholding: A refined edge map is produced by comparing pixel gradient magnitudes against both high and low threshold values, ensuring edge continuity while eliminating false edges caused by noise. Noise and illumination variation.

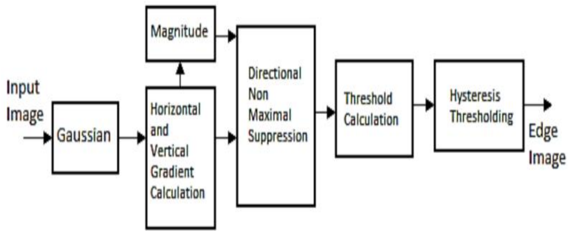


Figure1.1: Block Diagram Canny edge detection

II. LITERATURE REVIEW

In this section, the focus is on reviewing earlier research related to edge detection algorithms. Edge detection plays a crucial role in processing of images and computer vision, as it identifies locations in an image where intensity values change sharply. Among the various approaches, the Canny Edge Detection algorithm, proposed by John F. Canny in 1986, remains widely used due to its strong detection accuracy, precise localization, and reduced false responses. Through the years, significant research have been carried out to implement the Canny mask for edge detection on VLSI platforms such as FPGAs and ASICs, with the goal of achieving real-time processing, lower power consumption, and efficient hardware utilization. The computationally demanding nature of Canny's multi-stage process like smoothing, gradient calculation, non-maximum suppression, and double thresholding presents both opportunities and problems for hardware-based acceleration. Numerous studies have used creative algorithmic and architectural solutions to overcome these issues. Changes have been suggested in later research to boost the original Canny algorithm's effectiveness and versatility. The real-time implementation of Canny based edge detection on platforms such as FPGAs and GPUs has been made possible by developments in hardware acceleration. These advancements make it possible to apply the Canny approach to time-sensitive applications like robots, surveillance systems. Real-Time FPGA build with VGA Interface with input from a camera and output to VGA, complete Canny processing chain on a Spartan 3E FPGA, including the Gaussian filter, gradient, non-max suppression, and hysteresis. This demonstrates useful integration with vision systems. "Low Power and High-Speed Image Edge Detection Algorithm

Implementation on FPGA" by surveillance systems. Real-Time FPGA build with VGA Interface with input from a camera and output to VGA, complete Canny processing chain on a Spartan 3E FPGA, including the Gaussian filter, gradient, non-max suppression, and hysteresis. This demonstrates useful integration with vision systems. "Low Power and High-Speed Image Edge Detection Algorithm Implementation on FPGA" by Menaka, K.Deeba and R. Janarthanan focuses on optimizing performance for real-time applications by designing a fast, low-power edge detection technique on Fugato attain real-time processing performance, it focuses on hardware resource optimization. The design's ability to increase speed while decreasing power consumption makes it ideal for embedded systems, portable electronics, and applications that need quick and energy-efficient image processing. An approximation and adaptive Canny edge detector on an Intel Cyclone IV FPGA was proposed in another note worthy contribution. It employed Otsu's approach for adaptive thresholding with logarithmic approximations and reduced gradient computations. This solution reduced logic usage and maintained edge precision across a range of lighting situations, achieving a processing time of 1.231 ms per image at a 50 MHz clock frequency. For FPGA prototyping, the Canny edge detection approach with MATLAB is frequently used in conjunction with Simulink and HDL Coder. Scholars have investigated hardware-software co-simulation, Fuad and Rizvi used Spartan-6 FPGA to compare Sobel and Roberts operators. Others, such as Lin Bai, created FPGA-compatible models using MATLAB's Vision HDL Toolbox. Classic Canny implementations, such as 2D and 3D extensions, are also available in MATLAB Central. These projects demonstrate MATLAB's adaptability for both hardware deployment and algorithm development, which makes it an effective tool for real-time image processing applications and quick prototyping. In conclusion, more flexible, low-power, and real-time solutions are now the norm for VLSI-based Canny implementations. Modern embedded vision systems contain high-level programming tools, adaptive thresholding, pipelining, and approximation, which have made it possible for these architectures to achieve their requirements.

III. PROPOSED METHODOLOGY

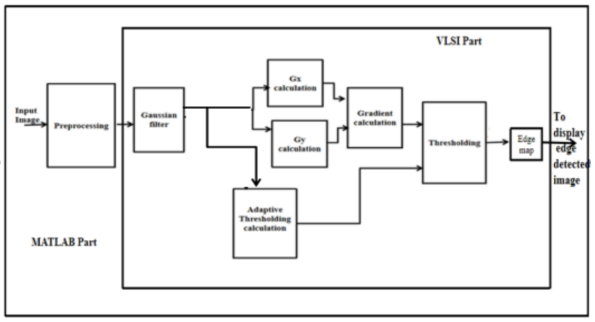


Figure 3.1: Adaptive threshold-based Canny edge detection

A. Preprocessing:

At this stage, the input image, whether in color or RGB format, is first resized and then converted into a grayscale image using MATLAB. The pixel values of the grayscale image are then extracted and stored in a text file.

B. Window Generation:

The pixel values from the text file are given as input for the pixel generator block and are converted into 3x3 image template having 9 values. The initial delay is the time required to fill the two FIFOs and nine shift registers.

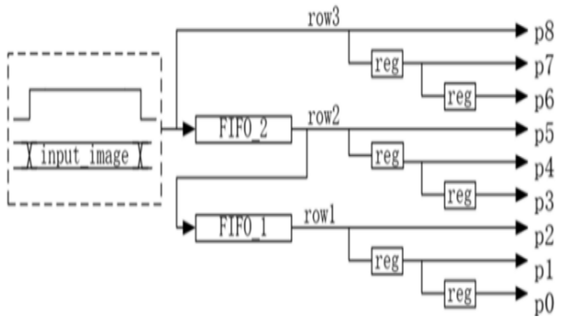


Figure 3.2: Window architecture(3x3 kernel)

C. Gaussian Filtering:

The grayscale image contains some noise in it so for removal of noise or unwanted signal from an image, Gaussian filter is preferred. Gaussian filter with size 3x3 is used to filter out the unwanted edges or signal. The filter is convolved with an image matrix to reduce the effect of false detection. Gaussian function for 2D is given below:

$$G(x,y)= \frac{1}{2\pi\sigma^2} exp\left(-\frac{x^2+y^2}{2\sigma^2}\right) \qquad (1)$$

where x and y represents horizontal values and vertical values of a 2D image, sigma is the standard deviation(σ) of the gaussian function increment of the value of σ means more blurred image hence reduce noise and provide smooth edge and G(x,y) represent the smoothened image.

Convolution Operation: Gaussian filter of 3x3 kernel matrix is convolved with sub matrix input pixel value.

$$\text{Gaussian filter} = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} * \begin{bmatrix} p0 & p1 & p2 \\ p3 & p4 & p5 \\ p6 & p7 & p8 \end{bmatrix}$$

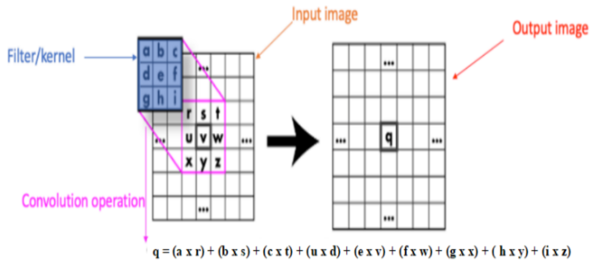

Input image
Output image
Filter/kernel
a b c
d e f
g h i
r s t
u v w
x y z
q
Convolution operation
q = (a x r) + (b x s) + (c x t) + (u x d) + (e x v) + (f x w) + (g x x) + (h x y) + (i x z)

Figure 3.3: Convolution Operation

D. Gradient Calculation:

The next stage involves gradient computation, where the results achieved from the Gaussian filtering unit are passed to the gradient block. In this step, the Sobel operator is employed for edge detection, utilizing two distinct kernel masks to determine the horizontal gradients and vertical gradients of the image.

(i) Horizontal sobel kernal: A kernel which approximate the intensity of pixel change in the x-direction.

(ii)Vertical sobel kernal: A kernel which approximate the intensity of pixel change in the y-direction.

$$\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \begin{bmatrix} g0 & g1 & g2 \\ g3 & g4 & g5 \\ g6 & g7 & g8 \end{bmatrix}$$

(a) Horizontal kernel (b)Vertical kernel (c)filtered pixel value

1. Horizontal and Vertical gradient Calculation: Convolution of filtered input pixel value with horizontal and vertical kernel respectively.

Horizontal gradient(Gx) = (g6 - g0) + 2(g7 - g1) + (g8 - g2)

Vertical gradient(Gy) = (g2 - g0) + 2(g5 - g3) + (g8 - g6)

2. Magnitude and direction calculation:

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

$$Direction(\theta) = tan^{-1}\left(\frac{Gx}{Gy}\right)$$

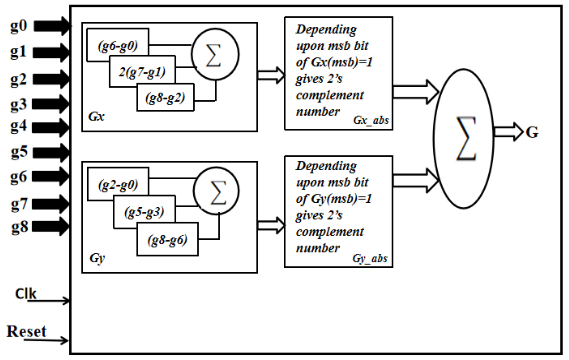

g0
g1
g2
g3
g4
g5
g6
g7
g8
Clk
Reset
(g6-g0)
2(g7-g1)
(g8-g2)
Σ
Gx
(g2-g0)
(g5-g3)
(g8-g6)
Σ
Gy
Depending upon msb bit of Gx(msb)=1 gives 2's complement number
Gx_abs
Depending upon msb bit of Gy(msb)=1 gives 2's complement number
Gy_abs
Σ
G

Figure 3.4: Architecture for Gradient Calculation

E. Adaptive Thresholding :

Image thresholding is a method used to separate dark and bright pixels according to their intensity levels. In the simplest form, known as fixed thresholding, each pixel value is compared against a predefined threshold. If the pixel's intensity exceeds the threshold, it is considered part of the edge. However, This approach tends to fail under varying intensity conditions within the image. To overcome this limitation, adaptive thresholding is applied, where the threshold is dynamically calculated to improve edge detection accuracy.

F. Edge mapping:

For each pixel look at the gradient direction. The pixel with the maximum gradient magnitude in its direction are only kept and other pixel gradient are suppressed or set to 0.Now, apply double thresholding i.e. high threshold(t_high) and low threshold(t_low) used to classifying pixels based on their gradient magnitudes as-

(i) *Strong Edges*: Pixel with gradient magnitude ≥ t_high are considered as strong edges.

(ii) *Weak Edges*: Pixel where t_low ≤ gradient magnitude < t_high are considered as weak edges.

(iii) *Non Edges:* Pixel with gradient magnitude < t_low are considered as non edges.

During the time of edge mapping, strong edges are always retained. Weak edges are only retained when they are linked to a strong edge pixel i.e. if the weak edge pixel is adjacent toward strong edge pixel, it is also considered as edge and those Weak edges not associated with strong edges are discarded as noise. It ensures continuity of edges.

IV. IMPLEMENTATION

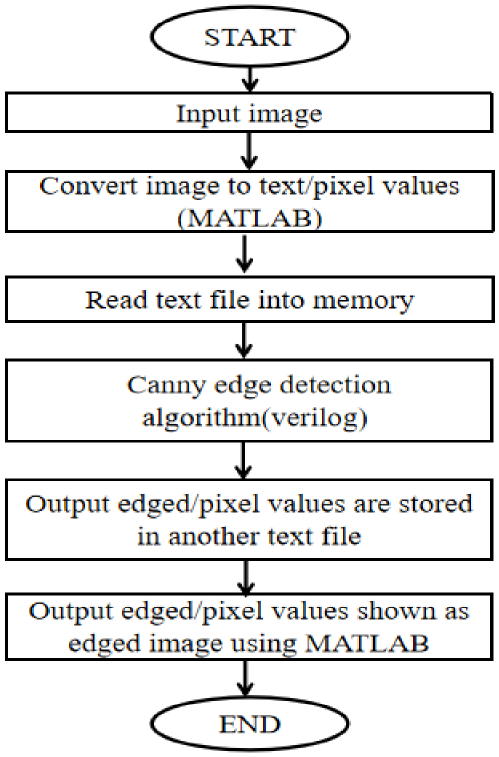

START
Input image
Convert image to text/pixel values (MATLAB)
Read text file into memory
Canny edge detection algorithm(verilog)
Output edged/pixel values are stored in another text file
Output edged/pixel values shown as edged image using MATLAB
END

Figure 4.1: Flow Chart of Proposed Methodology

1. Preprocessing of image using MATLAB

The flowchart of the proposed work is presented in Figure 4.1. In this process, the input RGB image is first converted into a grayscale image of resolution 256×256, with each pixel represented by 8 bits. The pixel values are then extracted and stored in a file using MATLAB, which is serving as a robust tool for mathematical modeling and simulation.

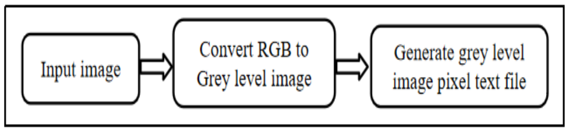

Input image
Convert RGB to Grey level image
Generate grey level image pixel text file

Figure 4.2: Preprocessing of input image

2. Hardware Implementation

The modified edge detection design in Verilog requires a text file containing grayscale pixel values, generated using MATLAB, given as the input for hardware implementation. This file act as the stimulus for the design during simulation. The modified Canny algorithm segmented into different stages, where each stage implemented as an independent Verilog module. In the testbench, the input text file is read, and the pixel data stored in a memory array is supplied to the input ports.

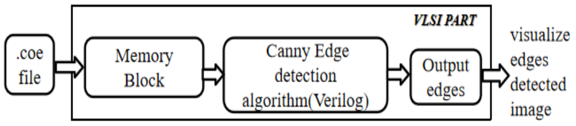

VLSI PART
.coe file
Memory Block
Canny Edge detection algorithm(Verilog)
Output edges
visualize edges detected image

Figure 4.3: Block Diagram Hardware Implementation

3.

4. Post-processing in MATLAB

Write the processed edge map to another text file and read this output text file to visualize the detected edges. Edge detected text/pixel values in the form on single column that will be converted into an image. These edged values are taken from the input image. The text/pixel values consist only two values, those are 255 it represent edge of an image and another value is 0 it represents the non edge of an image.

V. RESULT AND DISCUSSION

The proposed edge detection architecture for processing of image is designed by Verilog HDL and simulation and synthesis is performed in Vivado 2025.1 tool with target device Xczu7ev-ffvc11562e.

5.1 Pre-processing Phase

The RGB image on the left is transformed into the corresponding grayscale, and the pixel values are extracted using MATLAB, as depicted in Figure 5.1.1.

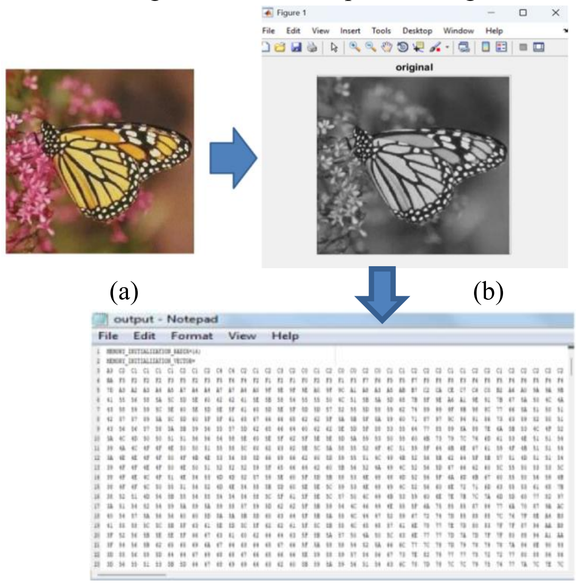


Figure 1
File Edit View Insert Tools Desktop Window Help
original
(a)
(b)
output - Notepad
File Edit Format View Help

(c)

Figure 5.1.1: (a) RGB Input Image (b) Greyscale image (c) Pixel values of image

5.2 Simulation Result of Adaptive Canny

After the preprocessing stage, the generated text file is used as input stimulus for the design during simulation. The text or pixel data is stored in memory, and all accesses are carried out from there. The edge detection architecture is described in Verilog HDL, along with a testbench, and simulation is performed to validate the functionality of the design. In VLSI development, simulation plays a critical role in verifying circuit behavior before fabrication. The general flow involves designing the circuit in HDL, creating the corresponding testbench, compiling both using a simulation tool, and then executing the simulation to analyze the output.

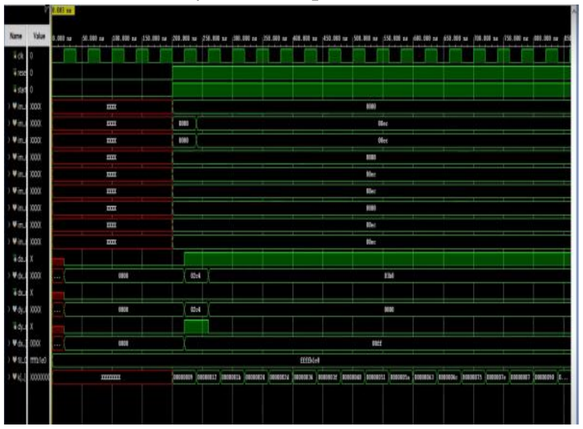


Figure 5.2.1: Simulation waveform

The edge detection with adaptive thresholding is executed with clock period 10ns and frequency of 100MHz.The simulation result waveform is illustrated in Figure 5.2.1.

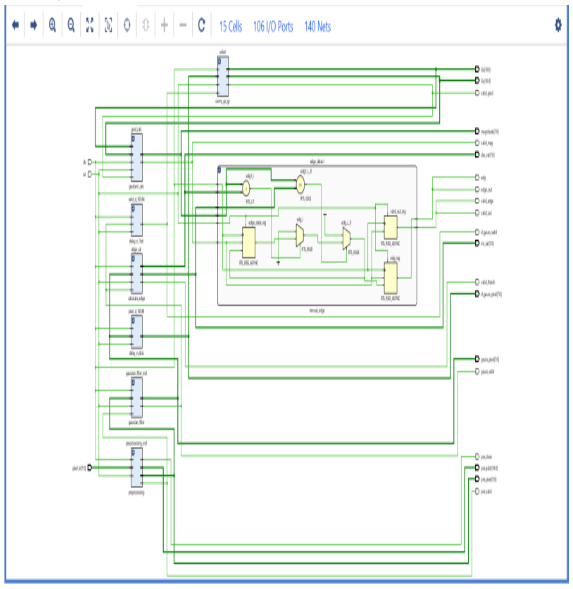


Figure 5.2.2: RTL Schematic

In VLSI design, an RTL(register transfer level) schematic represent circuit high level abstraction, focusing on the transfer of data between registers and the operation performed. It provide graphical representation of system data paths and control signals, illustrating how data is transferred and processed through functional block Figure 5.2.2.

5.3 Synthesis result Adaptive Canny

In Vivado, synthesis results shows gate level representation includes resource utilization (like count of LUTs, flip-flops, DSPs, I/O) and timing analysis (showing critical path, clock frequency).It suggest potential optimizations to improve performance and meet design constraints ensuring that the design is efficient and meets the required specifications for implementation.

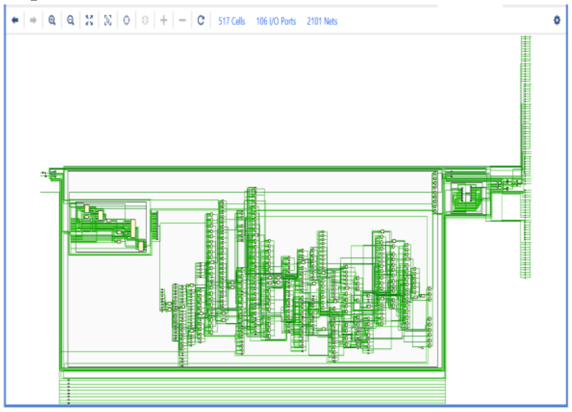
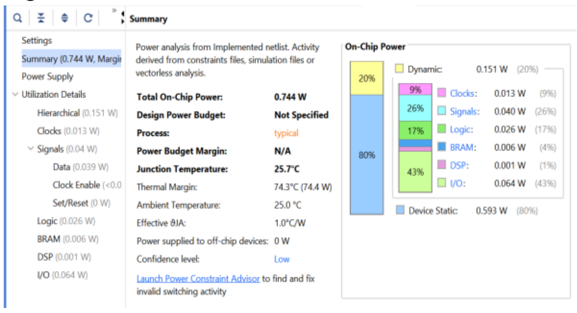


Figure 5.3.3: Technology Schematic

The synthesis results of the Verilog design present a detailed report on the conversion of the HDL description into its gate-level representation. The generated technology schematic consists of 517 cells, 106 input/output ports, and 2101 nets, as illustrated in Figure 5.3.3.



Summary

Settings
Summary (0.744 W, Margi
Power Supply
Utilization Details
Hierarchical (0.151 W)
Clocks (0.013 W)
Signals (0.04 W)
Data (0.039 W)
Clock Enable (<0.0
Set/Reset (0 W)
Logic (0.026 W)
BRAM (0.006 W)
DSP (0.001 W)
I/O (0.064 W)

Power analysis from Implemented netlist. Activity derived from constraints files, simulation files or vectorless analysis.

Total On-Chip Power:	**0.744 W**
Design Power Budget:	**Not Specified**
Process:	typical
Power Budget Margin:	**N/A**
Junction Temperature:	**25.7°C**
Thermal Margin:	74.3°C (74.4 W)
Ambient Temperature:	25.0 °C
Effective θJA:	1.0°C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Low

Launch Power Constraint Advisor to find and fix invalid switching activity

On-Chip Power

Dynamic:	0.151 W	(20%)	20%
Clocks:	0.013 W	(9%)	9%
Signals:	0.040 W	(26%)	26%
Logic:	0.026 W	(17%)	17%
BRAM:	0.006 W	(4%)	
DSP:	0.001 W	(1%)	
I/O:	0.064 W	(43%)	43%
Device Static:	0.593 W	(80%)	80%

Figure 5.3.4: Power Report adaptive canny

The power report reveals total on-chip power consumption of 3.338 watts, junction temperature of 25.7 degree Celsius. The report summary shows both static and dynamic power. Dynamic power is further analyzed by signals, logic, BRAM, DSP and I/O of board configuration, as shown in Figure 5.3.4.

5.4 Pre-processing Phase (MATLAB)

The output edged image obtained are shown in Figure 5.4.1 showing proposed methodology provide better result then traditional edge detection techniques.

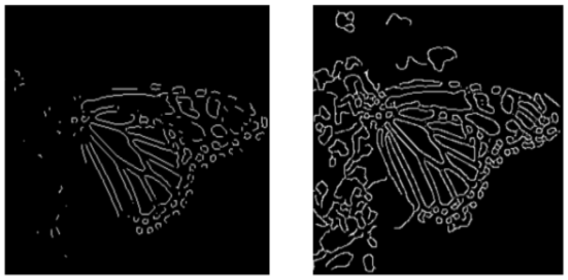


Figure5.4.1: (a) edged image canny (b) edged image adaptive canny

Table1:Comparison Synthesis Result of edge detector

Edge detection algorithm	Canny edge detection	Canny edge detection using adaptive thresholding
Image size	256x256	256x256
Target device	Xczu7ev-ffvc11562e	Xczu7ev-ffvc11562e
Dynamic Power	183mW	151mW
Static Power	630mW	593mW

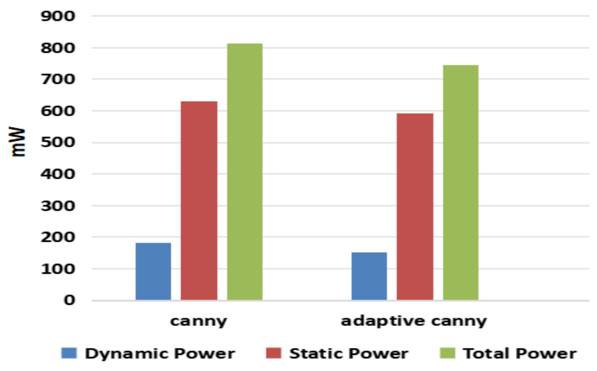


Figure 5.4.2: Power Analysis of edge detection

Table2:Comparison Reference paper with proposed adaptive edge detection

Reference	[2] (Soble)	Proposed edge detection
Target device	Spartan3E	ZCU104
Total number of 4 input LUTs	2369	1730
Number of slice register	1404	523
Power	------	744mW

VII. CONCLUSION

This work presents an FPGA-based implementation of edge detection that applies an adaptive thresholding approach to improve detection accuracy. The block-level edge detector with adaptive thresholding overcomes the drawbacks of conventional methods by reducing power consumption, execution time, and hardware area. The design is developed in Verilog HDL and evaluated through simulation and synthesis using the Xilinx Vivado tool. The proposed approach achieves low power usage, minimal area requirements, and reduced processing time, leading to lower latency and improved throughput.

VIII. FUTURE SCOPE

The FPGA-based adaptive edge detection system shows strong performance but offers scope for enhancement. Future extensions include support for high-resolution images and real-time video, enabling broader use in medical, surveillance, and navigation systems. Multi-scale detection can capture fine and coarse details, while machine learning can adapt thresholds under varying conditions. Energy-efficient FPGA–SoC designs, scalable reconfigurable modules, and integration into complete vision pipelines will further strengthen real-time image analysis applications.

REFERENCE

[1] Sanmugasundaram Ravichandran, Hui-Kai Su, Jui-Pin Yang, Dileepan Dhanasekaran, Manikandan Mahalingam, Wen-Kai Kuo "Parallel Processing of Sobel Edge Detection on FPGA: Enhancing Real-Time Image Analysis", Sensors 2025, Sensors 2025, Volume 25, Issue12, 3649. doi.org/10.3390/s25123649.

[2] Yaser Icer, Mustafa Turk "Implementation of Mainly Used Edge Detection Algorithms on FPGA", International Journal of Applied Mathematics, Electronics and Computers 2016,4,352-358.

[3] S.Neethu Raj, Alex.V "Parallel Block Based Architecture for Improved Edge Detection in Verilog" IJERT 2015, Volume 4, doi:10.17577/IJERTV4IS070778.

[4] Dr. Aziz Makandar, Shilpa Kaman, Rekha Biradar, Syeda Bibi Javeriya "Impact of Edge

Detection Algorithms on Different Types of Images using PSNR and MSE", LC International Journal of STEM 2023, Volume3, Issue4, pp. 1–11.doi:10.5281/ zenodo.7607059.

[5] Gaikwad, Vijay.N. Patil, "Low complexity illumination invariant motion Vector Detection Based on Logarithmic Edge Detection and Edge Difference". Fourth International Conference of Computing Communication Control and Automation (ICCUBEA), 2020.

[6] Dr. R.Menaka, Dr. R.Janarthanan, Dr. K.Deeba, "FPGA implementation of low power and high-speed image edge detection algorithm," Science Direct, 2020.

[7] P. Gupta, J.S. Kumar, U.P. Singh and R.K. Singh, ''Histogram Based Image Enhancement: A Survey",International Journal of Computer Sciences and Engineering, vol. 5, issue 6, (2017), pp.-177-182.

[8] S. Das, "Comparison of various edge detection technique", International Journal of Signal Processing, Image Processing and Pattern Recognition, vol.9, no.2, (2016), pp.143-158.

[9] E. Nadernejad, S. Sharifzadeh and H. Hassanpour, "Edge Detection Techniques Evaluations and Comparisons", Applied Mathematical Sciences, vol. 2, no. 31, (2008), pp. 1507 – 1520.

[10] R. Maini and H. Agrawal, "Study and Comparison of Various Image Edge Detection Techniques", International Journal of Image Processing (IJIP), vol. 3, issue 1, pp.1-12.

[11] P. P. Acharjya, R. Das and D. Ghoshal, "Study and Comparison of Different Edge Detectors for Image Segmentation", Global Journal of Computer Science and Technology Graphics & Vision, (2012), vol.12, issue 13, version 1.0.

[12] S. Kaur and I. Singh, "Comparison between Edge Detection Techniques", International Journal of Computer Applications, vol. 145, no.15, (2016), pp. 15-18.

[13] V. Saini and R. Garg, "A Comparative Analysis on Edge Detection Techniques Used in Image Processing", IOSR Journal of Electronics and Communication Engineering (IOSRJECE), ISSN: 2278-2834, vol. 1, issue 2, (2012), pp. 56-59.

[14] Ganesan, P.; Sajiv, G. A comprehensive study of edge detection for image processing applications. In Proceedings of the 2017 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS), Coimbatore, India, 17–18 March 2017; pp. 1–6. [Google Scholar] [CrossRef]

[15] Chaple, G.; Daruwala, R.D. Design of Sobel operator based image edge detection algorithm on FPGA. In Proceedings of the 2014 International Conference on Communication and Signal Processing, Melmaruvathur, India, 3–5 April 2014; pp. 788–792. [Google Scholar] [CrossRef]

[16] Sree, S.J.; Ashwin, S.; Kumar, S.A. Edge preserving algorithm for impulse noise removal using FPGA. In Proceedings of the 2012 International Conference on Machine Vision and Image Processing (MVIP), Coimbatore, India, 14–15 December 2012; pp. 69–72. [Google Scholar] [CrossRef]

[17] Khongprasongsiri, C.; Kumhom, P.; Suwansantisuk, W.; Chotikawanid, T.; Chumpol, S.; Ikura, M. A hardware implementation for real-time lane detection using high-level synthesis. In Proceedings of the 2018 International Workshop on Advanced Image Technology (IWAIT), Chiang Mai, Thailand, 7–9 January 2018; pp. 1–4. [Google Scholar]

[18] Mukhija, P.; Priyanka, P. Comparison of Different Edge Detection Techniques Used in License Plate Localization. In Proceedings of the 2022 Fifth International Conference on Computational Intelligence and Communication Technologies (CCICT), Sonepat, India, 8–9 July 2022; pp. 400–403. [Google Scholar] [CrossRef].

[19] Chaitra, M.; Aravind, H.S.; GR, A.S.; Bohara, H.; Shiak, N.A.; Srividhya, S. Design of Evaluation Board for Image Processing ASIC and VHDL Implementation of FPGA Interface. In Proceedings of the 2018 International Conference on Recent Innovations in Electrical, Electronics & Communication Engineering (ICRIEECE), Bhubaneswar, India, 27–28 July 2018; pp. 1284–1288. [Google Scholar] [CrossRef]

[20] Q. Xu, C. Chakrabarti, and L. J. Karam, "A distributed Canny edge detector: Algorithm and FPGA implementation," IEEE Transaction on image processing, Vol.23, No.7, July 2014.