International Scientific Journal of Engineering and Management (ISJEM) ISSN: 2583-6129
Volume: 04 Issue: 11 | Nov – 2025 DOI: 10.55041/ISJEM05201
An International Scholarly || Multidisciplinary || Open Access || Indexing in all major Database & Metadata

Real-Time Image Processing Acceleration Through Verilog HDL on FPGA Platforms

Sumanta Karmakar[1], Gurjeet Singh[1], Apurba Chatterjee[2], Souvik Auddya[3], Raj Vardhan Prasad[3]
[1]Assistant Professor, ECE Department, Asansol Engineering College
sumanta.ece@aecwb.edu.in, gurjeet.ece@aecwb.edu.in,
[2] Assistant Professor, EE Department, Asansol Engineering College
hodee@aecwb.edu.in
[3]Assistant Professor, EE Department, Asansol Engineering College

Corresponding author: sumantakarmakar799@gmail.com

Abstract
Real-time image processing is closely linked to monitoring systems that require immediate responses. In the medical field, it significantly enhances diagnostic technologies such as MRI and CT imaging. Likewise, modern security infrastructures rely on image processing for functions including facial recognition and surveillance. These cases demonstrate the wide-ranging applications and increasing importance of image processing across multiple domains.
The ISE Design Suite facilitates hardware development by providing tools for synthesis-based design compilation, timing analysis, RTL schematic generation, behavioral simulation under varying inputs, and device configuration through its integrated programming utilities.
In this work, design development and simulation are carried out using the Xilinx ISE Design Suite. Input images are first converted into hexadecimal format using MATLAB, ensuring smooth compatibility with the FPGA workflow. The outcomes clearly demonstrate the FPGA's capability to manage real-time image processing tasks with high efficiency and reliability.
Over time, the image processing field has evolved substantially. Early techniques were largely manual and focused on simple operations such as basic enhancement or image resizing. With advancements in technology, more sophisticated functions—like brightness control, inversion, thresholding, and binary conversion—have been refined to improve visual understanding. This project ultimately confirms that FPGA architectures offer a powerful and adaptable platform for implementing real-time image processing solutions.

Index Terms—*Xilinx ISE Design Suite, MATLAB FPGA, Image Processing.*

Introduction

In recent years, image processing has gained significant traction within embedded systems, particularly in applications such as real-time vision, medical diagnostics, and surveillance. Traditional software-based image processing solutions are often unsuitable for time-critical operations and low-power environments due to their high energy consumption and limited processing speed. As a result, Field Programmable Gate Arrays (FPGAs) have emerged as compelling hardware-oriented alternatives, offering high throughput, flexibility, and the ability to execute parallel data operations efficiently.
This paper outlines the design and implementation of essential image processing functions on an FPGA platform using Verilog Hardware Description Language (HDL). Employing HDLs provides designers with improved control over simulation, timing analysis, and the overall behavior of digital systems. The implemented algorithms leverage spatial parallelism to exploit the inherent parallel-processing capabilities of FPGAs.
As noted by Narayan, Jayamma et al. [2], this flexibility can be extended into various digital signal processing applications. Today, image processing finds widespread use in domains such as medical imaging,

meteorology, computer vision, digital photography, and microscopy [3]. Digital image processing frequently applies mathematical morphology to extract features from images. These nonlinear techniques enable modifications to pixel structures or the examination of geometric patterns, aligning with the definition of morphology—the study of shapes—described by Sumera and R. Ganesh [4].

Image enhancement techniques aim to improve the perceptual quality of an image either by emphasizing key features or reducing uncertainty between different regions [5]. This research demonstrates how input images can be preprocessed in MATLAB, converted into hexadecimal representation, and further processed using custom Verilog modules on an FPGA through the Xilinx ISE Design Suite. A wide array of digital image processing strategies is available, with the core algorithms in this work focusing on thresholding, inversion, contrast adjustment, and brightness control.

A. OVERVIEW

This work focuses on implementing fundamental image-processing operations on an FPGA platform using Verilog HDL, supported by MATLAB for image data conversion. MATLAB is primarily used to preprocess the input image, convert it into grayscale, and export it as a hexadecimal (.hex) file. This file serves as an interface that allows Verilog-based FPGA modules to efficiently access and manipulate pixel information.

Key image-processing functions—such as binary conversion, thresholding, brightness adjustment, and image inversion—are designed as individual Verilog modules. These modules are verified using Xilinx ISE simulation tools, and their accuracy and performance are evaluated by reconstructing and analysing the processed image output. The methodology highlights modularity, reusability, and real-time capability of the processing units. The main goal is to execute real-time operations, including brightness increment, brightness reduction, inversion, and thresholding on 24-bit RGB images. After development within the Xilinx ISE Design Suite, the final design is deployed onto an FPGA for hardware validation. This hardware-centric approach demonstrates the advantages of FPGA-based image processing, particularly its ability to deliver high-speed, parallel data handling for embedded vision applications.

B. OBJECTIVES

The importance of this work stems from its potential to improve image quality across numerous domains such as healthcare imaging, security surveillance, and remote sensing. For instance, automotive driver-assistance systems require real-time video analytics that traditional DSP processors cannot efficiently deliver due to limited processing power [6].

The objectives of this study include:

- Demonstrating real-time image processing using Verilog HDL through an FPGA-based implementation.
- Establishing smooth integration between MATLAB and Verilog by generating FPGA-compatible hexadecimal image data.
- Designing modular Verilog components to implement and test basic image-processing operations such as inversion, thresholding, and brightness control.
- Displaying and validating output results through simulation on the Xilinx ISE Design Suite.
- Investigating the capabilities of FPGAs to accelerate image processing compared to traditional software approaches.

Real-time image processing demands systems with characteristics such as high speed, adaptability, easy upgradability, and low development cost [7]. Tools like the Xilinx CORE Generator System provide optimized and customizable IP cores specifically built for Xilinx FPGA architectures [8].

C. APPLICATIONS

Using Verilog HDL on FPGA for image processing delivers significant benefits in applications where speed, determinism, and resource efficiency are essential.

- FPGA-Based Machine Learning Integration:

The scope of this work can be expanded by incorporating machine learning models for tasks like object detection or image classification. Deploying neural networks or lightweight ML models directly on hardware is highly advantageous for edge devices, and this project can serve as a starting point for such developments.

- Multichannel Image Processing:

Future enhancements may include multi-channel (RGB or multispectral) image-processing capabilities for advanced analysis. This is particularly relevant for medical imaging systems (MRI, CT) and satellite-based remote sensing.

- Real-Time Video Surveillance:

Continuous, low-latency video analysis is vital in security-sensitive environments such as airports, banks, and public infrastructures. Software-only systems often struggle with high-resolution, high-frame-rate video streams. FPGA-based implementations enable fast processing for applications including movement detection, facial recognition, and anomaly identification.

- Industrial Automation:

FPGA-based image processing supports on-chip inspection tasks such as defect identification, barcode interpretation, and machine vision in manufacturing. It is also beneficial in robotics, autonomous vehicles, and UAVs, where rapid thresholding, grayscale conversion, and feature extraction are required.

- Military and Aerospace:

FPGAs are widely used in defense and aerospace systems for guided missiles, satellite imaging, and surveillance drones. They offer rugged, high-performance image processing essential for navigation, targeting, and real-time observation.

METHODOLOGY

A. BRIGHTNESS OPERATION

• Increasing Brightness

In the proposed FPGA-based image processing framework, brightness enhancement is achieved by modifying the intensity values of each pixel's RGB components using Verilog HDL. A control signal, **SIGN**, determines whether the system should increase or decrease brightness. When the signal indicates addition (SIGN == 1), the design enhances throughput by processing two horizontally adjacent pixels per clock cycle.

Pixel data is accessed from one-dimensional memory arrays—**org_R**, **org_G**, and **org_B**—using a linear address calculated as *(WIDTH × row + col)*, where WIDTH denotes the image number of columns. A constant **VALUE** is added to each color channel to increase brightness.

To avoid overflow, the computed values are compared against the maximum 8-bit limit (255). If the sum exceeds 255, saturation logic clips the output to 255; otherwise, the computed value is retained. This ensures that pixel values remain within legal image boundaries, preventing distortion. The brightness-addition module supports low-latency, real-time image processing by utilizing FPGA parallelism and resource efficiency.

Decreasing Brightness

Brightness reduction is implemented by subtracting a fixed **VALUE** from the RGB intensities of each pixel whenever the control signal **SIGN** is low. Using the same linear addressing scheme, the system fetches two adjacent pixels per iteration from the image memory arrays.

To avoid underflow (values dropping below 0), conditional checks ensure that negative results are clipped to 0. Otherwise, the subtracted value is preserved. This guarantees that all pixel intensities remain within the valid 0–255 range. Processing two pixels simultaneously improves execution speed and optimizes FPGA resource utilization. This hardware-based architecture is suited for real-time applications such as contrast reduction, low-light filtering, and adaptive brightness control.

B. THRESHOLD OPERATION

Thresholding is a key technique in image processing used to identify important regions by converting grayscale images into binary form. A fixed threshold divides pixel intensities into two categories: pixels with intensities above the threshold become white (255), while all others become black (0). This method is computationally simple and ideal for FPGA-based real-time systems.

Mathematical Basis

Given an RGB pixel, its grayscale intensity is computed as:

$$Gray = \frac{R + G + B}{3}$$

The binary output is obtained using:

$$Binary = \begin{cases} 255, & \text{if } Gray > T \\ 0, & \text{otherwise} \end{cases}$$

This logic effectively separates the image into foreground and background regions.

In the Verilog implementation, the thresholding logic is enclosed within a conditional macro (**ifdef THRESHOLD_OPERATION**). For each pixel at coordinates *(row, col)*, the grayscale value is calculated and then compared with the predefined threshold **T**. If the grayscale value exceeds **T**, all RGB channels are set to white; otherwise, to black. The adjacent pixel at *(row, col + 1)* is processed similarly in the following block.

C. BLACK AND WHITE (GRAYSCALE) OPERATION

Converting an RGB image to grayscale is a fundamental step in many image processing pipelines. This transformation reduces the 24-bit RGB representation (8 bits per channel) into a single 8-bit intensity value while preserving critical visual information and simplifying further analysis.

Mathematical Principle

Each pixel consists of R, G, and B components, and its grayscale value is computed by:

$$Gray = \frac{R + G + B}{3}$$

While weighted grayscale methods (which emphasize green) may produce more visually accurate results, the averaging technique is preferred for FPGA implementations due to its simplicity and low hardware cost.

The Verilog grayscale module is conditionally executed using the **ifdef BLACK and WHITE_OPERATION** directive. For each pixel at *(row, col)*, the module retrieves the components from **org_R**, **org_G**, and **org_B**, computes the average, and assigns this grayscale value to all color channels of the output pixel (DATA_R0, DATA_G0, DATA_B0). This effectively converts the pixel into its black-and-white equivalent.

D. INVERT OPERATION

Inversion is a simple, yet effective pixel-level transformation widely used in FPGA-based real-time applications, including feature extraction, edge highlighting, and visual enhancement. The Verilog module responsible for inversion is wrapped within the **INVERT_OPERATION** macro, ensuring that the code is executed only when inversion is enabled during synthesis or simulation.

The inversion logic operates on RGB values stored in arrays **org_R**, **org_G**, and **org_B**, indexed using the linear address *(WIDTH × row + col)*. The inversion formula applied to each color component is:

$$Inverted = 255 - Original$$

This produces the complementary color of each pixel. The results for the current pixel *(row, col)* are stored in DATA_R0, DATA_G0, and DATA_B0. The values for the next pixel *(row, col + 1)* are stored in DATA_R1, DATA_G1, and DATA_B1. This dual-pixel processing pattern is characteristic of FPGA architectures optimized for high-throughput image processing.

LITERATURE REVIEW

Image transformation processes often lead to quality degradation, making enhancement techniques essential for improving visual clarity. Methods such as contrast stretching, brightness modification, inversion, and thresholding are commonly applied to refine image quality. Although both software and hardware platforms can execute these operations, hardware-based solutions provide significantly higher performance. This work emphasizes the use of reconfigurable hardware—specifically Field Programmable Gate Arrays (FPGAs)—to achieve real-time image enhancement using Hardware Description Languages (HDLs). Such an approach introduces an efficient and flexible technique within digital system design and VLSI applications [9].

Thinning, another key image processing technique, extracts the structural skeleton of an object from a digital image. It identifies only the essential pixels that represent the object's shape, resulting in a simplified and compact representation of image information [10].

In recent years, programmable devices such as FPGAs, Complex Programmable Logic Devices (CPLDs), and Application-Specific Integrated Circuits (ASICs) have become increasingly prominent in digital image processing due to their speed and hardware-level optimization capabilities [11].

FPGAs, in particular, have established themselves as powerful platforms for high-speed image processing. Unlike digital signal processors (DSPs), which rely on sequential execution, FPGAs leverage customized, parallel hardware architectures for superior throughput. Research has explored FPGA-based implementations of enhancement algorithms, including brightness control, contrast manipulation, negative imaging, thresholding, and filtering. Using MATLAB's System Generator, a modular image-processing framework was created and deployed on a Spartan-3E board. Experimental evaluation demonstrated improvements in visual

quality and efficient hardware resource usage, highlighting System Generator's potential for real-time FPGA algorithm development [12].

The Bitmap (BMP) format is frequently used in such studies due to its raw, uncompressed nature, which preserves complete pixel information. While this results in large file sizes, BMP is ideal for applications that require pixel-level accuracy and do not tolerate compression distortions [13].

Further research highlights the increasing demand for high-performance image-processing methods that have traditionally relied on software environments like MATLAB. A notable study implemented four basic enhancement operations—thresholding, contrast adjustment, brightness modification, and image inversion—using Verilog HDL on a Terasic DE0-Nano FPGA platform. The results demonstrated superior performance, reduced execution time, and improved long-term efficiency compared to pure software solutions. The study confirms the effectiveness of hardware acceleration in improving processing accuracy and system responsiveness, while also offering a scalable design for future enhancements [14].

Another work proposes an effective method for improving images degraded by low lighting and haze, conditions that significantly impact computer vision performance. The authors developed a simple yet powerful enhancement algorithm combining brightness and contrast adjustments, implemented entirely in Verilog HDL for hardware efficiency. Testing on 100 vehicle license plate images showed that their method outperformed two conventional approaches in terms of peak signal-to-noise ratio (PSNR) and mean square error (MSE). The study suggests strong potential for FPGA-based prototyping in real-time systems such as automated license plate recognition and surveillance applications [15].

From a storage perspective, JPEG (JPG) uses lossy compression to significantly reduce file size, making it suitable for web applications and bandwidth-limited environments. However, repeated editing degrades quality due to irreversible data loss. Conversely, BMP retains full pixel integrity and is thus preferred for precise operations such as FPGA-based processing, editing, printing, and algorithm evaluation. In this work, both formats are used depending on the requirements—JPEG for compact storage and BMP for high-fidelity processing. This dual-format strategy contributes to the distinctiveness of the project.

The versatility of the presented design lies in its ease of modification. Parameters such as brightness level, threshold value, or image dimensions can be changed with minimal adjustments to the Verilog code, allowing the same architecture to support a wide range of image-processing tasks. The FPGA implementation is well suited for real-time applications, compatible with multiple Xilinx families, and supports on-chip memory for continuous, high-speed image testing.

These advantages collectively demonstrate the strong potential of FPGA-based image processing using Verilog HDL within the Xilinx ISE Design Suite, reinforcing its relevance for modern embedded vision applications.

SOFTWARE IMPLEMENTATION

The software implementation of this project relies on two primary tools: **MATLAB** for preprocessing the input image and **Xilinx ISE Design Suite** for simulating and executing Verilog-based image processing algorithms. Each software component plays a vital role in preparing and processing the image before it is transferred to the FPGA.

A. Image Preprocessing Using MATLAB

The first stage involves preparing the input image in MATLAB so it can be efficiently processed by the FPGA hardware. The source image, *project_image_99.jpg*, is imported using MATLAB's imread() function. The image has a resolution of **512 × 768 pixels** and is processed in its original **RGB format** rather than being converted to grayscale. This allows direct manipulation of pixel-level values across all three color channels.

For each pixel, MATLAB extracts the **red, green, and blue** components and stores them consecutively in a one-dimensional array. To ensure FPGA compatibility, the 8-bit RGB values are converted into **hexadecimal** format. Following the row-major convention required by the Verilog modules, the image is traversed **from the bottom row to the top row**.

MATLAB's file-handling commands — fopen, fprintf, and fclose — are used to write the processed pixel values to a **.hex file**, placing one hexadecimal entry per line. Upon completion, a confirmation message appears in the MATLAB console.

This HEX file serves as the input for the Verilog testbench, which subsequently performs operations such as negative transformation, brightness variation, thresholding, and binary conversion.

This preprocessing step ensures that the MATLAB-generated image data is fully compatible with the FPGA environment, enabling smooth integration with the Verilog-based hardware modules.

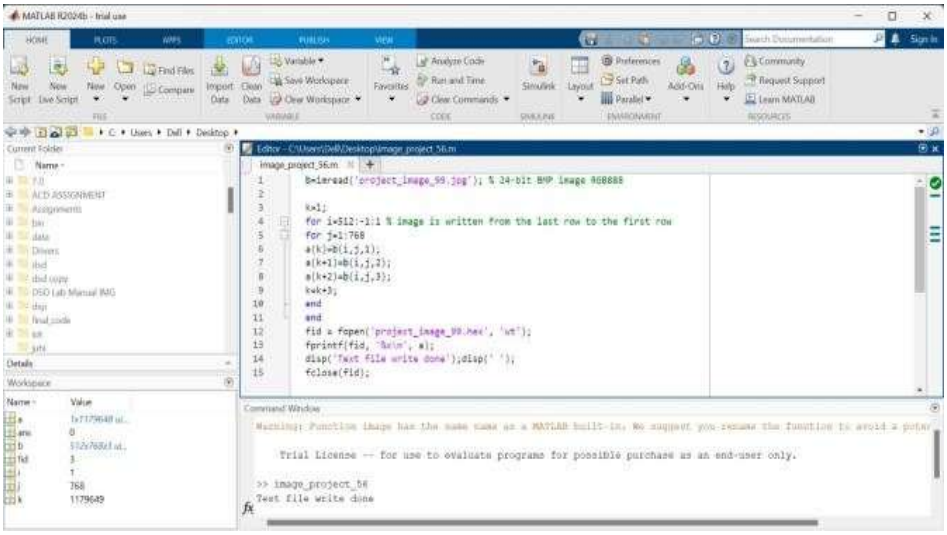

Figure 1 MATLAB CODE WINDOW

International Scientific Journal of Engineering and Management (ISJEM) ISSN: 2583-6129
Volume: 04 Issue: 11 | Nov – 2025 DOI: 10.55041/ISJEM05201
An International Scholarly || Multidisciplinary || Open Access || Indexing in all major Database & Metadata

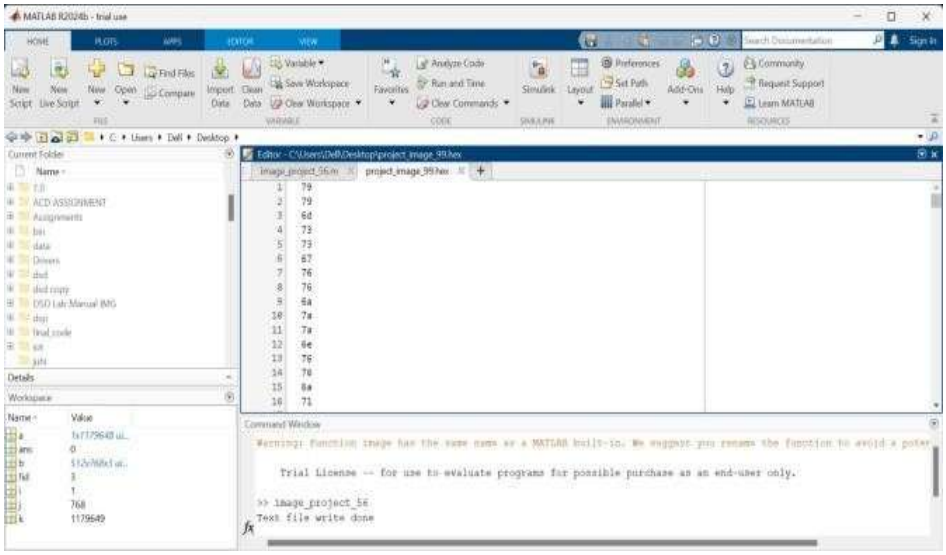

Figure 2 MATLAB HEX FILE WINDOW

From the figure 1 we can see that code for the jpg to hex file is implemented and our input file "project_image_99.jpg"

Is successfully converted into hex file as we can see it in figure 2.

A. Simulation and Processing in Xilinx ISE Design Suite

The hardware development environment, the Xilinx ISE Design Suite, is used to implement, simulate, and synthesize the Verilog HDL code. The testbench downloads the MATLAB-generated .hex file and positions the pixel intensity values into FPGA memory structures. One of the image processing functions—inversion, brightness addition/subtraction, thresholding, or black-and-white conversion—is carried out in response to control signals.

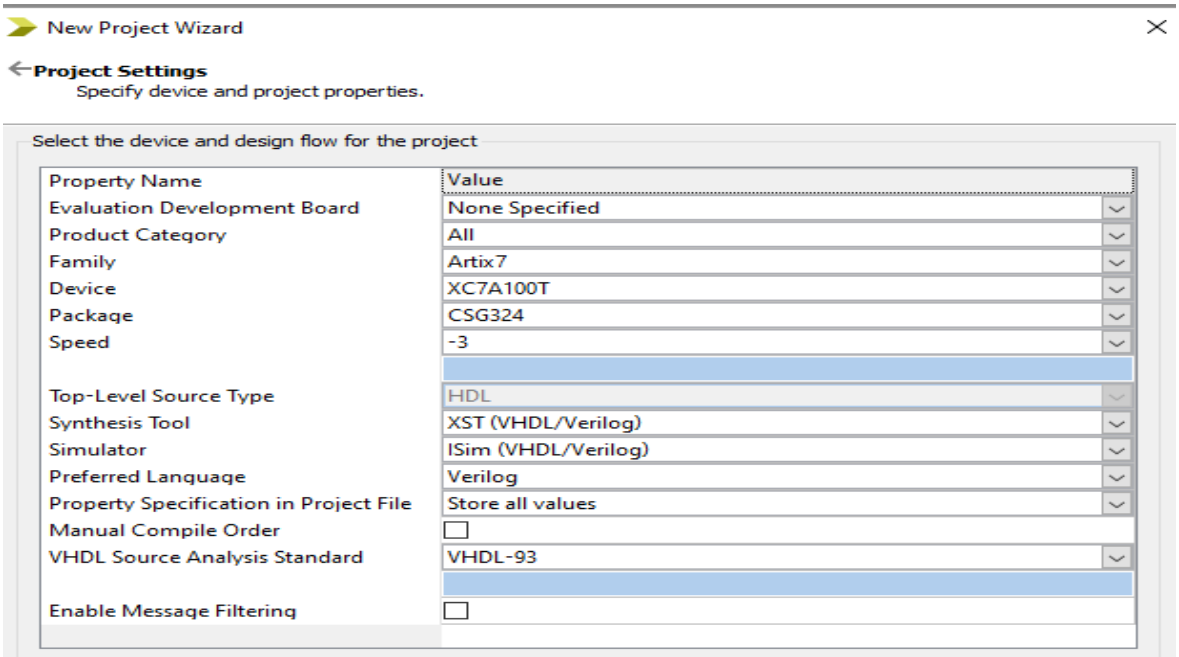

Figure 3 FPGA BOARD SELECTION

For the implementation of the following above image processing functions, we have to select the FPGA board.

We can see the figure 3 where we have selected the FPGA board of Artix 7 family and the device was XC7A100T and the package is CSG324 with a speed of -3.

In **Figure 4**, the hierarchy structure plays a crucial role. The parameter file appears at the top of the hierarchy since it contains all the include files as well as the necessary define statements. Below it is the testbench, which is further divided into two submodules: **image_read** and **image_write**. Maintaining this order is essential for correct behavioral simulation. The hierarchy also displays the project files created by the user along with the name of the selected FPGA board.

After processing, the Verilog testbench writes the modified pixel data into a new bitmap (.bmp) output file. This allows the processed image to be viewed directly, helping verify the correctness of the Verilog implementation. All operations—from image reading to output generation—are executed entirely inside the ISE environment, accurately emulating how the design would function on actual FPGA hardware. No additional MATLAB post-processing is required.

Once the desired operation is chosen (by uncommenting the corresponding module), the simulation can be executed. To generate the simulation window, the user selects the testbench and runs the simulation.

Figure 5 displays the simulation output window. To complete the simulation, the user enters **run 6 ms**, waits for execution to finish, closes the window, and then locates the generated output file in the project directory.

The resulting image is saved automatically in the ISE folder. Depending on the requirement, the output can be generated in either **BMP** or **JPG** format.

RESULTS

The proposed image processing framework was implemented successfully by using MATLAB for input preprocessing and the Xilinx ISE Design Suite for simulating the Verilog HDL–based hardware modules. Since Verilog cannot directly read JPG files, the input image was first converted into BMP format. Five different image operations—inversion, brightness enhancement, brightness reduction, black-and-white conversion, and thresholding—were executed by enabling the corresponding block within the parameter file. This approach demonstrates the modularity and flexibility of the design, as each operation can be activated simply by uncommenting its respective section.

Figure 7(a) shows the original 768×512 RGB image, which serves as the reference for all processing outputs. MATLAB extracted the RGB pixel values and stored them in a .hex file, which is then read by the Verilog testbench.

The inverted output image shown in Figure 7(b) is generated by applying the transformation 255 – pixel_value to every 8-bit pixel, effectively reversing the intensity levels. This inversion logic is implemented directly in Verilog.

Brightness enhancement, shown in Figure 7(c), is achieved by adding a constant scalar value to each pixel. Any result exceeding the 8-bit maximum is capped at 255. The processed image exhibits visibly increased illumination, particularly in darker regions, without causing excessive saturation.

Conversely, brightness reduction—displayed in Figure 7(d)—subtracts a constant value from each pixel and clamps the output at 0 to avoid underflow. This produces a darker version of the input image, useful in scenarios requiring contrast suppression or low-light simulation. The output confirms that bright regions are appropriately dimmed while maintaining image structure.

The black-and-white image in Figure 7(e) is generated by converting the input into a binary format using a predefined threshold. This operation evaluates each pixel and assigns it either 0 or 255, producing a strictly two-level representation.

Figure 7(f) illustrates the thresholded output, which further enhances contrast and suppresses high-frequency content. This fixed-level thresholding technique effectively highlights prominent edges and removes background noise, making it suitable for feature extraction and object detection applications. Thresholding essentially performs a special form of quantization where pixel values above the threshold are set to 255 and those below are set to 0, as commonly applied in 8-bit grayscale images [16].

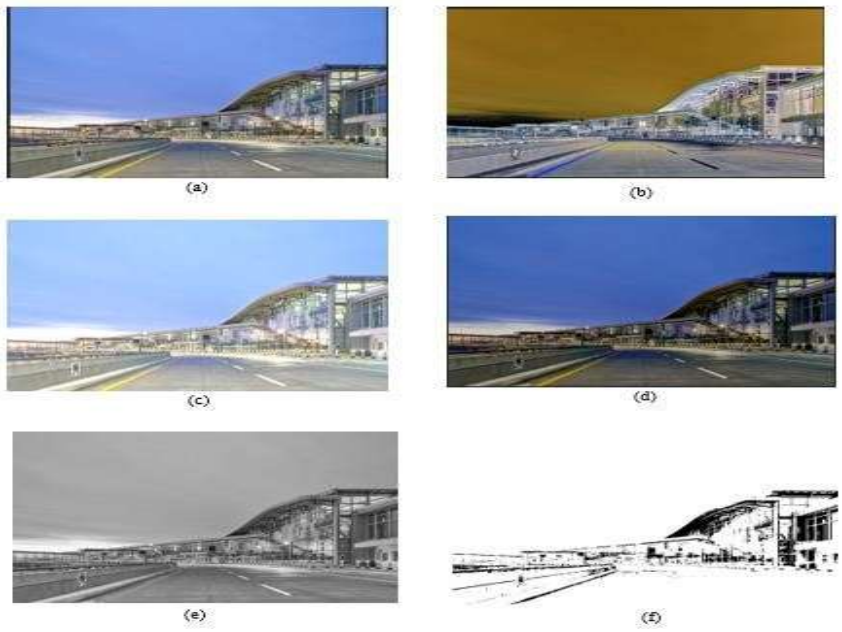

Figure 4 (a) ORIGINAL IMAGE (b) INVERTED IMAGE (c) ADDITION BRIGHTNESS IMAGE (d) SUBTRACTION BRIGHTNESS IMAGE (e) BLACK & WHITE IMAGE (f) THRESHOLD IMAGE

FUTURE SCOPE

• Incorporation of Advanced Image Processing Techniques:

Future enhancements may include implementing more sophisticated operations such as object detection, image sharpening, morphological transformations, edge detection methods (like Sobel and Canny), and filtering algorithms (such as Gaussian and median filters). Adding these will significantly expand the capability of FPGA-based imaging systems, enabling applications such as motion tracking and facial recognition.

• Hardware Acceleration for Machine Learning:

The framework can be extended to accelerate convolutional neural networks (CNNs), which are widely used for real-time tasks such as object classification and human detection. Implementing neural network blocks on FPGA would greatly enhance processing speed and efficiency.

• Real-Time Video Stream Processing:

Transitioning from static image processing to continuous video processing would unlock applications in autonomous navigation, surveillance systems, and medical imaging. The inherent parallelism of FPGAs makes them ideal for such real-time demands.

• Integration with IoT and Edge Devices:

The architecture can be adapted for deployment in IoT-based systems or edge processors, enabling on-device decision-making in smart surveillance, industrial monitoring, and automated inspection systems.

• Support for High-Resolution Imaging:

Extending the design to handle high-resolution images (4K, 8K) while preserving real-time performance would test the scalability of the system and further highlight the efficiency of FPGA-based solutions.

• SoC-Based Implementation:

The design may be ported to System-on-Chip platforms such as Xilinx Zynq, which integrate an FPGA fabric with an ARM processor. This would support hybrid image-processing workflows combining hardware acceleration with software control.

• Direct Camera Interface:

Instead of relying on pre-stored images, the FPGA can be connected to live camera modules to process real-time video feeds. This enhancement would support applications such as autonomous robotics, industrial automation, and intelligent traffic systems.

• FPGA–Cloud Collaboration:

The architecture could be extended to push processed outputs to cloud servers for further analysis or storage. This would be highly beneficial for IoT ecosystems and remote monitoring applications.

• Multichannel and Multispectral Image Processing:

Future implementations can incorporate multi-channel (RGB) or multispectral image processing to enhance accuracy. This is especially valuable in medical diagnostics (MRI, CT), agriculture, and satellite imaging.

CONCLUSION

This work presents a hardware-accelerated approach to executing essential image processing operations using Verilog HDL on an FPGA platform within the Xilinx ISE Design Suite. The implemented functions—including brightness adjustment, inversion, thresholding, and brightness reduction—were successfully realized, illustrating the FPGA's strength in handling pixel-level parallelism. MATLAB played a vital role in converting JPG images into HEX format, simplifying the data preparation stage and enabling smooth integration with the hardware modules.

The overall hardware–software co-design highlights the performance advantages of FPGAs over traditional software-only solutions, particularly in terms of speed, configurability, and resource optimization. The results confirm the feasibility of using FPGAs for real-time image transformation and provide a solid foundation for future enhancements such as edge detection, filtering, and AI-assisted image analysis.

ACKNOWLEDGMENT

I extend my heartfelt gratitude to my project guide, Prof. Parul Panchal, for their continuous support, insightful feedback, and valuable guidance throughout this work. Their expertise in image processing and FPGA systems has been instrumental in completing this project successfully.

References

- Nada Qasim Mohammed, Muataz H. Salih, Rafikha Aliana, Qasim Mohammed Hussein and Noor Aldeen A. Khalid," FPGA implementation of multiple image processing algorithms using spatial parallelism" presented ARPN Journal of Engineering and Applied Sciences vol. 13, no. 15, august 2018.
- Prof. Narayan A. Badiger, Miss.Jayamma Muragod, Miss.Poornima Pattar,Miss. Priyanka Gundlur, Miss.Savita Bhadri, "FPGA Implementation of Image Enhancement using Verilog HDL" presented International Journal of Engineering & Technology (IRJET) Volume 7 Issue 6, June 2020.
- Neha. P. Raut, Prof.A.V.Gokhale, "FPGA Implementation for Image Processing Algorithms Using Xilinx System Generator " presented IOSR Journal of VLSI and Signal Processing (IOSR-JVSP) Volume 2, Issue 4 (May. – Jun. 2013)
- Sumera Sultana, R.Ganesh , "FPGA Implementation of Binary Morphological Processing for Image Feature Extraction" presented International Journal of Engineering & Technology (IRJET) Volume 4 Issue 10, October 2015.
- Dr. Sagar Patel, Krinesh Patel, Keval Patel and Chaitanya Patel, "Image Enhancement on FPGA using Verilog " presented International Journal of Technical Innovation in Modern Engineering & Science (IJTIMES) Volume 5 Issue 3, March 2019.
- Shamsiah Suhaili, Joyce Huong Shing Yii, Asrani Lit, Kuryati Kipli, Maimun Huja Husin, Mohamad Faizrizwan Mohd Sabri, Norhuzaimin Julai, "Development of Digital Image Processing Algorithms via FPGA Implementation " presented Semarak International Journal of Electronic System Engineering Volume 3 Issue 15, September 2024.
- iuliana chiuchișan, marius cristian cerlincă," Implementation of Real-Time System for Medical Image Processing using Verilog Hardware Description Language" presented Recent Researches in Medicine, Biology and Bioscience.
- B.Muralikrishna, K.Gnana Deepika, B.Raghu Kanth, V.G.Swaroop Vemana , "Image Processing using IP Core Generator through FPGA " presented International Journal of Computer Applications Volume 46– No.23, May 2012
- Mahavir Singh, Gitanjali Pandove, "An Implementation of Image Enhancement on Real Time Configurablse system using HDL" published in International Journal of Advanced Research in Electronics and Communication engineering volume 7, issue 3, March 2018.
- Dr G Sankar , "Implementation of an Image Thinning Algorithm using Verilog and MATLAB " presented Journal of Nonlinear Analysis and Optimization Volume 13(9) (2022), September 2022.
- Babu Ravi Teja, Abhilash S. Warrier, Akshay S. Belvadi, Dhiraj R. Gawhane, "Design and Implementation of Neighborhood Processing Operations on FPGA using Verilog HDL" presented

IOSR Journal of VLSI and Signal Processing (IOSR- JVSP) Volume 4, Issue 1, Ver. II (Jan. 2014).

- Kalyani A. Dakre, “A Review on Image Enhancement using Hardware co-simulation for Biomedical Application" presented International Journal of Advanced Research in Computer Engineering & Technology (IJARCET) Volume 3 Issue 12, December 2014.

- Sri Lakshmi , Mery Lavanya , Nikhilesh Mohan, Naga Pavan , Kavya Reddy,"HDL implementation for real- time image properties adjustments" presented International Journal of Engineering & Technology (IRJET) Volume 11 Issue 4, April 2024.

- Mandeep Singh Narula, Nishant Singla, "FPGA Implementation of Image Enhancement Using Verilog HDL" presented International Research Journal of Engineering and Technology (IRJET) Volume 05 Issue 05,May 2018.

- Zul Imran Azhari, Samsul Setumin, Emilia Noorsal, Mohd Hanapiah Abdullah, "Digital image enhancement by brightness and contrast manipulation using Verilog hardware description language" presented International Journal of Electrical and Computer Engineering (IJECE) Volume 13 Issue 2, April 2023.

- Kankanala Vanitha, D. Sampath Kumar, "FPGA Implementation of Image Processing Using Image Enhancement Algorithms " presented International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering(IJAREEIE) Volume 7, Issue 2, February 2018.