As per UGC guidelines an electronic bar code is provided to secure your paper

International Journal for Modern Trends in Science and Technology
Volume 12, Issue 03, pages 348-359.
ISSN: 2455-3778 online
Available online at: http://www.ijmtst.com/vol12issue03.html
DOI: https://doi.org/10.5281/zenodo.19037903

Implementation of Image Algorithms using Verilog HDL

Bhavani[1] | Hima sree[2] | Manoj kumar[2] | Shabana[2] | Leela Krishna[2]

[1]*Associate Professor, Department of Electronics & Communication Engineering, SR Gudlavalleru Engineering College, Gudlavalleru, Andhra Pradesh, India.*
[2]*Department of Electronics & Communication Engineering, SR Gudlavalleru Engineering College, Gudlavalleru, Andhra Pradesh, India.*

To Cite this Article
Bhavani, Hima Sree, Manoj Kumar, Shabana & Leela Krishna (2026). Implementation of Image Algorithms using Verilog HDL. International Journal for Modern Trends in Science and Technology, 12(03), 348-359. https://doi.org/10.5281/zenodo.19037903

Article Info
Received: 10 February 2026; Revised: 09 March 2026; Accepted: 12 March 2026.

KEYWORDS

Image Processing, Grey scale conversion, brightness manipulation, thresholding, contrast modification, Inversion, smoothing, Verilog

ABSTRACT

The main theme of this paper is to construct on an FPGA platform a number of widely used digital image processing algorithms, including thresholding, inversion, contrast modification, brightness manipulation, and grey level translation. To prepare the input colour image for FPGA processing, it is first transformed to bitmap format and then, using MATLAB, to hexadecimal format. The algorithms are developed in Verilog HDL, and the simulation and synthesis platforms are Model Sim Altera and Intel Quartus II. Therefore, five hex files are created, each of which represents the image that was processed using a certain algorithm. After that, MATLAB post-processes these outputs to recreate and display the finished pictures.

INTRODUCTION

The goal of this project is to provide image processing algorithms that are Compatible with Field Programmable Gate Arrays (FPGAs). SDG 3, which encourages health and well-being, is supported by using this technology in medical imaging. FPGAs increase image processing efficiency, which improves diagnostic precision and leads to better health outcomes. Semiconductor devices known as FPGAs can be programmed after they are manufactured, enabling customization for a range of uses. Because of its architecture's capability for parallel pipelining, operations are completed more quickly and efficiently, leading to better performance.

Computer-assisted manipulation and analysis of digital photographs is known as digital image processing. The approach uses L2I algorithms and statistical models to extract useful information and enhance image quality. Greyscale conversion, contrast enhancement, brightness modification, thresholding, inversion ,smoothing are important methods.

RELATED WORK

This book[1] provides an accessible introduction to Field Programmable Gate Arrays (FPGAs), focusing on their architecture, design flow, and practical applications.

The authors explain how FPGAs differ from microprocessors and ASICs, highlighting their flexibility, parallelism, and ability to accelerate computing tasks. The Intel® special edition emphasizes FPGA integration with Intel processors and tools, outlining design entry methods (HDL, high-level synthesis) and application domains such as signal processing, networking, embedded systems, and hardware acceleration. The text is written in a simplified style, making it suitable for beginners, while also giving insights into industry practices, advantages, and limitations of FPGA-based solutions. This [2]widely cited textbook is a foundational reference in the field of image processing. It covers the theoretical basis and practical techniques for image enhancement, restoration, compression, segmentation, and representation. The authors present both spatial and frequency domain approaches, with a strong emphasis on mathematical modeling and algorithm development. The book has been extensively used in academic and research contexts to provide a comprehensive understanding of digital image processing fundamentals, making it a cornerstone reference for studies involving image analysis and computer vision. Vanaparthy [3] presented real-time hardware image enhancement techniques utilizing FPGA technology. Their work emphasized implementing various image enhancement algorithms such as brightness control, contrast stretching, negative transformation, thresholding, and filtering techniques on FPGA platforms.

A technique for image enhancement based on FPGA was presented by Chiuchisan and Geman [4] with the goal of enhancing skin cancer diagnosis using computer-aided diagnosis systems. To improve image quality and extract vital information for a precise diagnosis, digital image processing is necessary in medical imaging. Digitisation of images, compression, quality improvement, restoration, matching, description, and reconstruction are examples of important facets of digital image processing. Commonly used algorithms to improve photographs of skin lesions or tumours include contrast correction, brightness modification, grey level transformation, inversion, and thresholding. Zhang [5] used a cheap OV7670 camera to create a color-to-grayscale converter. A Video Graphics Array (VGA) interface was used to process and display the captured images. Modules such as Camera Controller, Image Capture, and VGA Master for FPGA integration were developed using Verilog HDL.

The design and development of a GPU specifically suited for image processing applications were covered by Panappally and Dhanesh [6]. A four-stage pipelining procedure that includes instruction fetching, decoding, processing pixel data, and returning the processed output to memory is supported by the APU architecture. Using MATLAB and Verilog HDL, Chaithra [7] concentrated on real-time image improvement. Their investigation demonstrated the benefits of FPGA-based image processing over DSPs, mainly because of HDL's parallel processing capabilities. The image-enhancement algorithms were synthesised with Xilinx XST, simulated with ISim from Xilinx ISE Design Suite 14.3, and implemented on FPGA. Before being fed into Xilinx ISE for processing, input bitmap images were transformed into hexadecimal representation using MATLAB.

Dhanabal [8] investigated using Verilog HDL to modify brightness, threshold, and contrast in digital images for coin counting and circle recognition. FPGA-based hardware image processing is renowned for being affordable and reconfigurable, which makes it appropriate for improving image quality. Nived [9] used FPGA and Verilog HDL to study real-time programmable picture enhancement. They highlighted Verilog HDL's parallel architecture, which grants a clear edge over MATLAB and DSP-based solutions by facilitating quicker processing on FPGA hardware.

To speed up real-time license plate recognition, Azhari [10] used image enhancing techniques to hardware. Bitmap images had to be converted into hexadecimal format using MATLAB. After being processed by a Brightness and Contrast Adjuster block and saved into an Output Image Memory block, each RGB pixel value was reconverted to bitmap format. Chiuchisan [11] used Verilog HDL and FPGA to create a real-time, programmable picture enhancing system. Through hardware-level image processing procedures such edge recognition, sharpening, and contrast and brightness adjustment through the creation of specialised filter series, the study concentrated on enhancing the quality of medical images.

Vision Systems Design [12] compared CPUs and FPGAs for image processing, highlighting that FPGAs provide lower latency and higher parallel performance, while CPUs offer easier programmability. Raikovich and Fehér [13] applied partial reconfiguration of FPGAs in image processing, demonstrating flexible hardware adaptation for real-time applications. Keerthi et al. [14] implemented image enhancement algorithms using Verilog HDL, showing FPGA-based acceleration of brightness and contrast adjustment. Khudhair et al. [15] proposed a color-to-grayscale conversion method based on singular value decomposition, achieving high-quality image preservation. Greyscale scanning has advantages in certain applications, like medical imaging, because it requires less storage and improves diagnostic clarity, according to a report by Electronic Office Systems [16]. Documentation by Terasic Technologies [17] described the DE10-Standard FPGA board layout, outlining resources for hardware implementation of digital systems.

Guide by UNLV [18] introduced Altera Quartus II software, explaining FPGA design flow and synthesis for hardware projects.

Tutorial by Mentor Graphics [19] provided practical instructions on using ModelSim for simulation and verification of HDL designs. MathWorks [20] explained MATLAB as a computational platform widely used for numerical analysis, algorithm development, and image processing research. Yusefi et al.'s study [21] demonstrated the bio-mediated synthesis of magnetic nanoparticles using fruit peel extract, emphasizing the manufacture and characterization of environmentally friendly nanomaterials.

This paper main goal is to provide digital image processing algorithms for FPGA implementation using Verilog HDL. Greyscale conversion, brightness manipulation, thresholding, contrast modification, and inversion ,smoothing are the suggested algorithms that are employed in the project. For many image processing applications, including improving image quality, extracting significant information, and adjusting images for varied display circumstances, these techniques are essential. Their versatility in real-time video processing, surveillance systems, satellite imaging, and medical imaging make them significant. Both ModelSim Altera and Intel Quartus II are utilised in this project. The design is synthesised and implemented onto the FPGA using Intel Quartus II, while the Verilog HDL-based design is simulated using ModelSim Altera to confirm both ModelSim Altera and Intel Quartus II are utilised in this project. The design is synthesised and implemented onto the FPGA using Intel Quartus II, while the Verilog HDL-based design is simulated using ModelSim Altera to ensure the design's functionality. Prior to being processed by various image processing algorithms in Verilog HDL, the input image is first translated into hexadecimal format. The image data is then subjected to several image processing methods, including inversion, thresholding, contrast modification,brightnessmanipulation, smoothing and greyscale conversion. The image data that has been processed is then exported to the appropriate hex files. The hexadecimal files are converted to Portable Network Graphics (PNG) format for the processed output images using Matrix Laboratory (MATLAB).

IMAGE PROCESSING

The process of reducing an image's RGB dimensions to the intensity value is known as greyscale conversion. A greyscale image's pixel values vary from 0 (black) to 255 (white). Because a greyscale image uses less memory and processes more quickly than a colour image, this algorithm is utilised. For instance, this method is frequently employed in medical imaging procedures like CT, MRI, and X-rays. Due to the excellent resolution of the greyscale image, even the slightest anomalies may be precisely identified, providing a thorough glimpse of the body's interior components.

A colour image can be converted to a greyscale image using one of three techniques. The first approach, referred to as the lightness method, uses Eq. (1) to compute grey scale values by averaging the RGB data's minimum and maximum $\text{Gray}=(\text{Min}(R,G,B) + \max(R,G,B))/2$ (1)

The main problem with this approach, though, is that it doesn't make advantage of the RGB components, which are three colours. The second approach, referred to as the average method, uses Eq. (2) to average the RGB data values to produce the greyscale values. Even though our eyes see colour differently, this approach has a problem in that it assigns the same weight to each RGB data point.

$$\text{Gray}= (R+G+B)/3 \qquad (2)$$

The third approach, referred to as the luminosity method, uses a weighted combination of the RGB data to determine the greyscale values. The green channel is given the most weight in this strategy, followed by the red and blue channels. The second approach is used in this project.

Reading the incoming colour image and dividing it into three distinct channels (red, green, and blue) is the first step in the greyscale conversion process. As demonstrated in Eq. (3), the greyscale data is produced by adding up the RGB data of the same pixel and dividing the result by three. The result is a grey output image.

$Gray = 0.2989 \times R + 0.587 \times G + 0.114 \times B$ (3)

One of the fundamental functions of digital picture processing is brightness manipulation. To improve visibility, we would want to boost the brightness of the input when it is very light. On the other hand, adding brightness is essential if the image is too drak to increase user visibility. In essence, brightness modification is the process of adding or removing a fixed value from each image pixel. Each pixel in the image can have a value added to it to improve brightness, and vice versa. The first step in implementing brightness alteration is to read the greyscale image's data. Both brightness addition and subtraction brightness data are derived using the corresponding mathematical formulas shown in Eqs. (4) and (5). The output image of brightness grey is produced.

$Brightness{=}Gray{+}x$ (4)

$Brightness{=}Gray{-}x$ (5)

Through thresholding, a greyscale image is converted to a binary image based on a predetermined threshold value. The final image is exclusively in black and white. In computer vision applications like object detection and monitoring, it is essential. It can help with automatic tracking and recognition by separating the objects from the background with the right threshold value. It facilitates analysis and judgement. According to Eq. (6), the new threshold value will be set to 255 (white) if the initial grey value is greater than x, and 0 (black) if it is less than or equal to x. The threshold value that establishes whether an output pixel is black or white is denoted by the variable x.

The first step in applying thresholding is to read the greyscale image's data. The mathematical expression shown in Eq. (6) can be used to obtain threshold data for thresholding. It yields the threshold output image.

$255(white)\ if\ Gray > x$

$0\ (black)\ if\ Gray \leq x$ (6)

By altering contrast, one can modify the intensity difference between the image's pixels and improve the image. The bright pixels will get brighter, and the dark pixels will get darker as the contrast is increased. Each image pixel can be multiplied by a contrast factor to accomplish this function. Contrast increases when contrast factor > 1 and decreases when contrast factor < 1. In order to guarantee that the text and information are readable and clear, it is frequently utilized in optical character recognition (OCR) systems and digitization initiatives.

This improves the overall document interpretation and text extraction accuracy. The mathematical expression for contrast adjustment, where x is the contrast factor, is displayed in Equation (7). The first step in implementing contrast adjustment is to read the greyscale image's data. The contrast output image is created by applying the mathematical expression shown in Eq. (7) to calculate the contrast data.

$Contrast = x \times Gray$ (7)

The process of inversion entails flipping each pixel's intensity values. Consequently, the bright pixels will turn dark, and the dark pixels will get bright. In medical imaging, inversion helps make specific features or structures in pictures more visible. In X-ray imaging, for instance, inverting the image might draw attention to specific features that might not be as noticeable in the original. This aids the physicians in making a more thorough and precise diagnosis and analysis. The maximum pixel value for a grayscale image is 255. The new Invert value can be obtained by subtracting the original grey value from the maximum pixel value, Amax, using the mathematical expression in Eq. (8). The first step in implementing inversion is to read the greyscale image's data. The mathematical expression displayed in Eq. (8) can be used to retrieve the inverted data. The produced image is inverted.

$Invert{=}Amax{-}Gray$ (8)

A image processing method called a smoothing filter, sometimes called a mean filter, is used to lessen noise and slight intensity changes in digital images. It functions by substituting the average value of nearby pixels for the value of a single pixel. This procedure maintains the

image's overall structure while lowering high-frequency elements like randomnoise.
This project uses Verilog HDL for greyscale image processing and implements a 3 x 3 mean filter. The filter works on a nine-pixel sliding window, substituting the average of all the pixels in the window for the central pixel.

METHODOLOGY

Researching and studying data pertaining to digital image processing techniques is the initial stage. Numerous relevant papers and research projects have been examined and reviewed.

Finding appropriate image processing hardware and software is the next stage. For instance, the FPGA development tools and methodology for writing Verilog HDL code. The Intel Quartus II and ModelSim Altera are the platforms chosen, and the FPGA board utilised is the DE-10 Standard. Finding an appropriate image for digital image processing is therefore essential.

Since Lenna.png is a common test image in image processing, it was selected. When reading and storing the picture data, the Verilog code specifies the image's dimensions, which must be known in advance. The Lenna.png file has dimensions of 512 x 512 x 3. The image is then converted from PNG to bitmap format using Microsoft Paint. The bitmap image is converted to hexadecimal format using MATLAB similarly skull.png and weather.png has dimensions of 512 x 512 x 3. The image is then converted from PNG to bitmap format using Microsoft Paint. The bitmap image is converted to hexadecimal format using MATLAB because the FPGA can only accept binary and hexadecimal files. In order to

$$\text{Pixel}_{out} = \frac{1}{9}\sum_{i=1}^{9}\text{Pixel}_i$$

read the bitmap image and convert it to hexadecimal representation, a MATLAB script is created. In order to read the bitmap image and transform it into hexadecimal format, a MATLAB script is created. Intel Quartus II and ModelSim Altera are then used to input the hexadecimal file into the FPGA.

To create and execute various digital image processing algorithms on the hexadecimal data of the input image, multiple Verilog HDL codes are written. To confirm that the design works, a testbench is created. Simulation was carried out once the Verilog HDL code was synthesised. To create the corresponding graphics, MATLAB post-processes the hexadecimal files that the algorithms produce. The data in the hexadecimal files is converted into PNG format using a different MATLAB algorithm.
Intel Quartus II and ModelSim Altera are then used to input the hexadecimal file into the FPGA. To create and execute various digital image processing algorithms on the hexadecimal data of the input image, multiple Verilog HDL codes are generated. To confirm that the design works, a testbench is created. Simulation was carried out once the Verilog HDL code was synthesised. The algorithms produce hexadecimal files, which are then post-processed in MATLAB to produce the corresponding images. To convert the data in hexadecimal files into PNG format, another MATLAB script is created. The generated output photos are examined to make sure they live up to expectations. The Verilog HDL code is changed, and the simulation is run again if the output images do not match expectations.

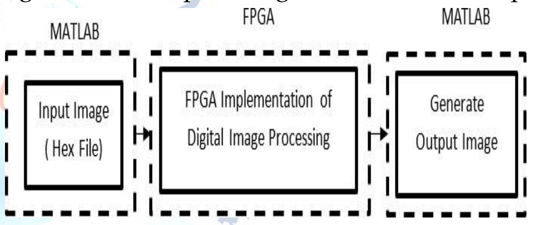


Fig1: Block diagram for the FPGA implementation of digital image processing algorithm

Figure 1 shows the block diagram for the FPGA implementation of the digital image processing algorithm. A 512 x 512 x 3 colour image called Lenna.png is selected as the input. It is transformed into a bitmap format using Microsoft Paint, producing Lenna.bmp. MATLAB is then used to convert it to hexadecimal representation. Similarly, 512 x 512 x 3 colour image called skull.png and weather.png are selected as the inputs. It is transformed into a bitmap format using Microsoft Paint, producing skull.bmp and weather.bmp. MATLAB is then used to convert it to hexadecimal representation.

Using Intel Quartus II and ModelSim Altera, the FPGA reads the RGB data from the input image and stores it in the memory block. A variety of digital image processing techniques are developed using Verilog HDL. Greyscale conversion, brightness manipulation, thresholding, contrast modification, and inversion are among the image processing procedures that are applied to the input image. The hexadecimal files containing the processed data are produced by the algorithms. MATLAB, which transforms the image data from hexadecimal format back to PNG format, is then used to generate and visualise the treated images.

RESULTS AND DISCUSSION:

This part covered the Verilog coding needed to create different digital image processing algorithms as well as the MATLAB code used to convert the colour image into hexadecimal representation. After that, MATLAB is used to post-process the output hexadecimal files back into images. This section discusses the specifics of the image's pre-processing. Prior familiarity with the FPGA board, Intel Quartus II, ModelSim, and MATLAB is required in order to use the software efficiently.

Preprocessing using the Microsoft Paint:

Digital image processing frequently uses the Lenna picture, a standardised test image. The Lenna image that was downloaded from the Internet was first saved as Lenna.png in PNG format. The image's dimensions are 512x512x3. Microsoft Paint is then used to transform the image into bitmap format. Lossless compression, which effectively lowers the image file size without compromising image quality, is a feature of the PNG format. However, the bitmap format, which stores the image data pixel-by-pixel, is renowned for its lossless and uncompressed qualities. This makes bitmap format images the preferred format for image processing activities since each pixel's colour is recorded in a matrix of bytes. Simarly we can format the images like skull.png, weather.png. The below figures 2(a),2(b),2(c),2(d),2(e),2(f) shows the png and bmp formats of lenna ,weather, skull.

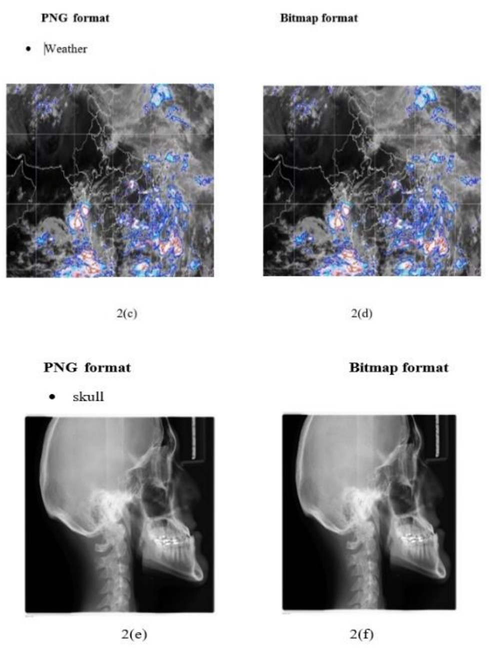


Fig2: PNG and Bit map images

Preprocessing Using MATLAB :

Images must be converted from bitmap to hexadecimal format because the FPGA cannot read images directly. Only binary and hexadecimal picture data can be read by FPGA. In this project, the bitmap image's pixel values are converted into a hexadecimal data file using MATLAB.

MATLAB Coding Explanation :

Image has a size of 512 x 512 x 3 and is in color. This indicates that the image has three channels (Red, Green, and Blue) and is 512 wide by 512 high. After that, it transforms the image's RGB values into the appropriate hexadecimal representations. The hex data will be converted to an image in post-processing using the image size file. The Lenna , skull, weather image has three color channels, or RGB channels, because they are colour images. Each pixel in the image's color information is represented by these values. The hexadecimal data file is subsequently read by Quartus II's Verilog code for FPGA-based digital image processing.

Output of hex files:

1.Lenna Image:

E2 →R0
89→G0
7D→B0
E2→R1
89→G1

85→B1
DF→R2
88→G2
80→B2
B3→R262141
46→G262141

2.Skull

FD →R0
FD→ G0
FD →B0
FE →R1
FE→G1
FE →B1
FD →R2
FD →G2
FF→R262141
FF→G262141
FF→B262141

3.Weather

7A→R0
7A →G0
7A →B0
78→R1
78 →G1
78 →B1
65→R2
65 →G2
65→B2
56→G262142
59→B262142
41→R262143

Image Processing using Intel Quartus II

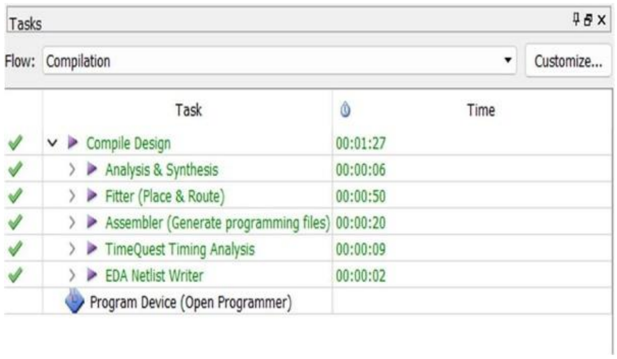
Quartus II compiles the design. Compilation includes analysis and synthesis, fitter, assembler, timing analysis, and Exploratory Data Analysis (EDA) netlist writer. Flip-flops and logic gates are used to convert the Verilog code into a gate-level representation once it has been checked for grammatical errors. The result is an RTL design that shows the logic circuits and data flow. The ports are mapped to the correct pins in the Pin Assignment so that data can enter and exit the FPGA.

Verilog coding explanation :

Verilog code is divided into testbench and design sections. The top module, gray module, brightness module, threshold module, contrast module, and invert module are the six modules that make up the design portion. To create an RTL design and program it into the FPGA hardware, the Verilog coding for the design in Quartus II needs to be synthesizable. The RGB data from the input image file is processed once every clock cycle thanks to the Verilog coding. The functioning of the top module is checked using the testbench file design_module_tb. It lists all of the wires, registers, parameters, and integers utilized by the test bench that is displayed in the blue box. The clk and reset signals displayed in the orange box are initialized. The rgb hex data is read from image and stored in the in box, which is indicated in the red box. For the top-level module to communicate with the testbench environment, it instantiates the top_module and establishes a connection with the signals in the testbench.

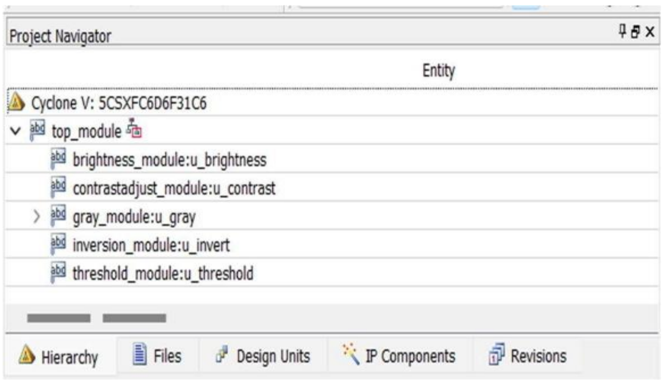
Compilation report :

top_module is the main module, with gray_module,brightness_module,threshold_module,contrast_module,andinvert_module as its submodules. fig 3(a) shows the compilation task and time ,fig3(b) shows the project navigator, fig3(c) shows the compilation report.


Tasks
Flow: Compilation | Customize...

	Task	Time
✓	Compile Design	00:01:27
✓	Analysis & Synthesis	00:00:06
✓	Fitter (Place & Route)	00:00:50
✓	Assembler (Generate programming files)	00:00:20
✓	TimeQuest Timing Analysis	00:00:09
✓	EDA Netlist Writer	00:00:02
	Program Device (Open Programmer)	

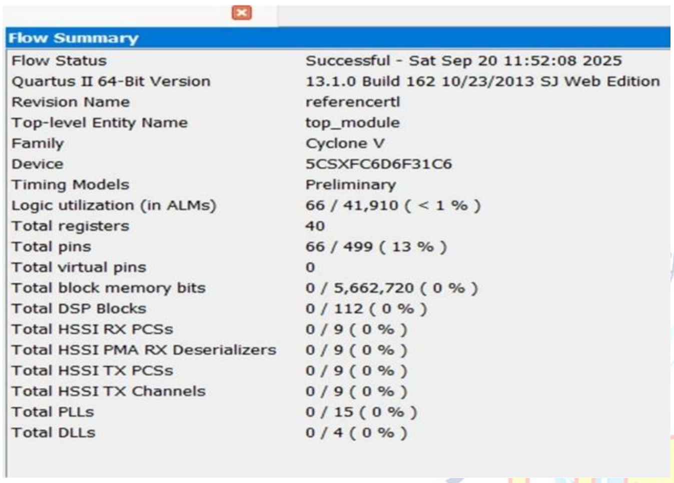
Fig3(a):Compilation tasks and time


Project Navigator

Entity

Cyclone V: 5CSXFC6D6F31C6
- top_module
  - brightness_module:u_brightness
  - contrastadjust_module:u_contrast
  - gray_module:u_gray
  - inversion_module:u_invert
  - threshold_module:u_threshold

Fig3(b) :project navigator


Flow Summary

Flow Status	Successful - Sat Sep 20 11:52:08 2025
Quartus II 64-Bit Version	13.1.0 Build 162 10/23/2013 SJ Web Edition
Revision Name	referencertl
Top-level Entity Name	top_module
Family	Cyclone V
Device	5CSXFC6D6F31C6
Timing Models	Preliminary
Logic utilization (in ALMs)	66 / 41,910 (< 1 %)
Total registers	40
Total pins	66 / 499 (13 %)
Total virtual pins	0
Total block memory bits	0 / 5,662,720 (0 %)
Total DSP Blocks	0 / 112 (0 %)
Total HSSI RX PCSs	0 / 9 (0 %)
Total HSSI PMA RX Deserializers	0 / 9 (0 %)
Total HSSI TX PCSs	0 / 9 (0 %)
Total HSSI TX Channels	0 / 9 (0 %)
Total PLLs	0 / 15 (0 %)
Total DLLs	0 / 4 (0 %)

fig 3(c) : compilation report

RTL Desgin:

The RTL design describes how the digital circuit functions by outlining the data flow between the registers and the logical operations performed on the data. It is an essential step in the hardware development process since it acts as a guide for the synthesis and implementation phases. RTL designs are created using Verilog code after Quartus has finished compiling. The Top Module's RTL schematic, top_module. It provides a visual representation of the hierarchical structure, showing how data moves between different components. It also shows how the inputs, outputs, internal wires, registers, and submodules are connected in the architecture. It highlights the integration and contributions of multiple submodules to the overall image processing pipeline.

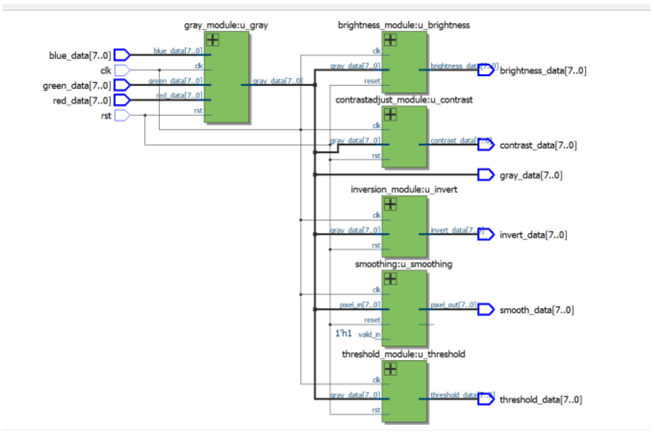


Fig4.RTL design for top_module

FPGA Implementation :

Prior to implementation, certain steps need to be taken to ensure the design's operation and FPGA integration. Pin Planner allocates the proper pins to the input and output ports. The programmer detects the FPGA hardware and uploads the Verilog code into the FPGA.Pin Planner is used to map the design's input and output ports to the FPGA device's physical pins. This is to ensure that the FPGA and the external signals and components are properly communicating. The pin assignment is found in the DE10-Standard User Manual. The cl is given to CLOCK_50, which has a clock input of 50 MHz.

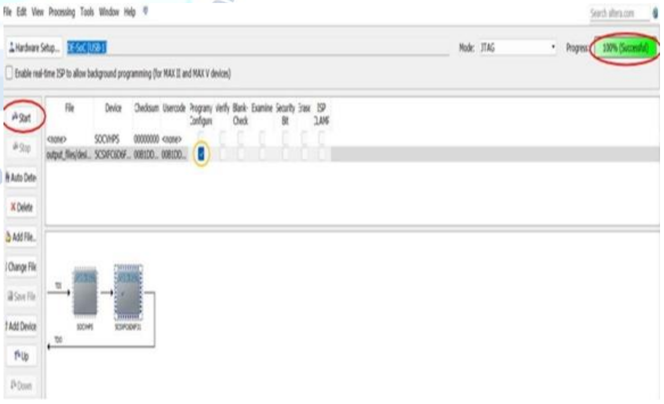


Fig. 5. Programmer interface for FPGA programming of the sof file

The FPGA is set up in JTAG mode in the programmer. The option DE-SoC [USB-1] is selected under Hardware Setup. After that, the device 5CSXFC6D6 and the button Auto Detect are chosen. The Programmer interface displays both the FPGA and HPS. To switch to 5CSXFC6D6F31, right-click on the FPGA. The output file top_module.sof is selected when the FPGA is right-clicked once again to modify the file. In Figure 5, the.sof file is downloaded into the FPGA by clicking the Start button and selecting the Program/Configure box. The Verilog code has been successfully uploaded into the FPGA board, as indicated by the 100% (Successful) message.

When Verilog coding is applied to the FPGA then the data that the FPGA has recorded for Lenna ,Weather, Skull are shown in 6(a),(b) and (c) respectively. When a stream of input image data enters the FPGA, the six 7-segment displays record the red, green, and blue data of E2,98, and 7d, respectively. The LEDs average the recorded red,green, and blue data to produce the binary number 10000011 of the gray_data for Lenna and similarly for weather FA,FA,FA and for skull 7A,7A, and 7A. The connection between the laptop and the FPGA board is shown in Figure 5, along with the laptop's monitor indicating the status of the software upload into the FPGA.

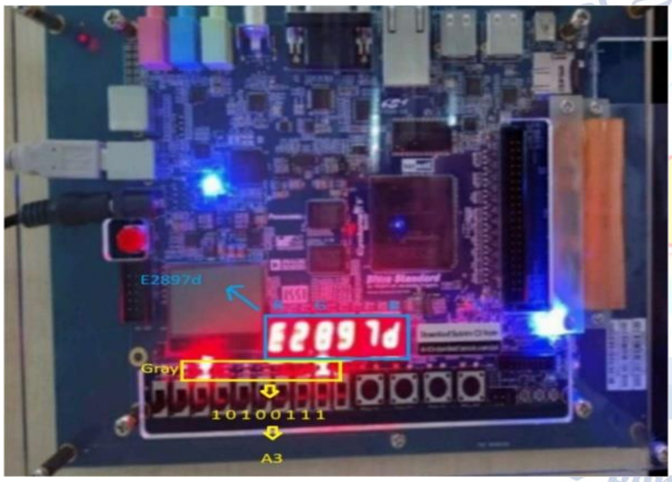


Fig 6(a): The information obtained by the FPGA for the Lenna

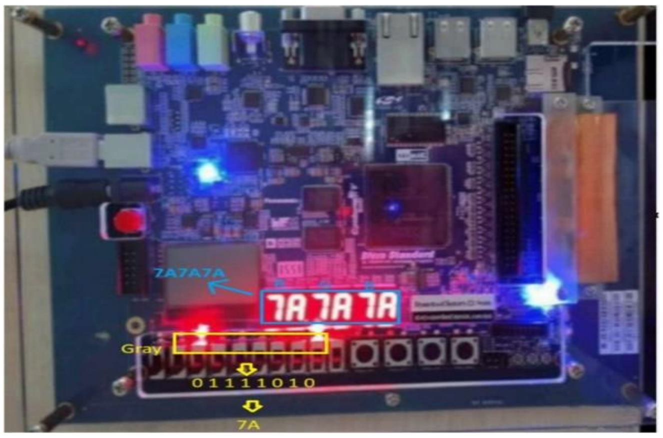


Fig 6(b):Information obtained by the FPGA for the Weather

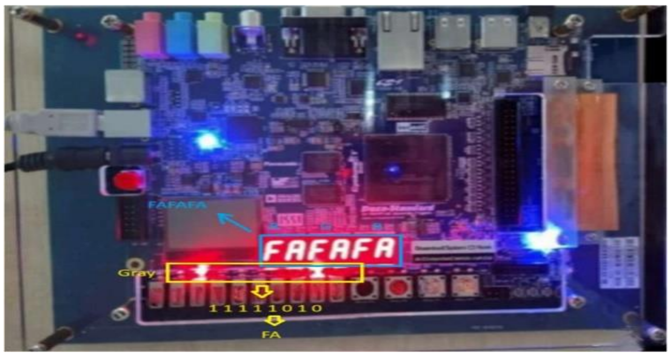


Fig 6(c):The information obtained by the FPGA for the skull

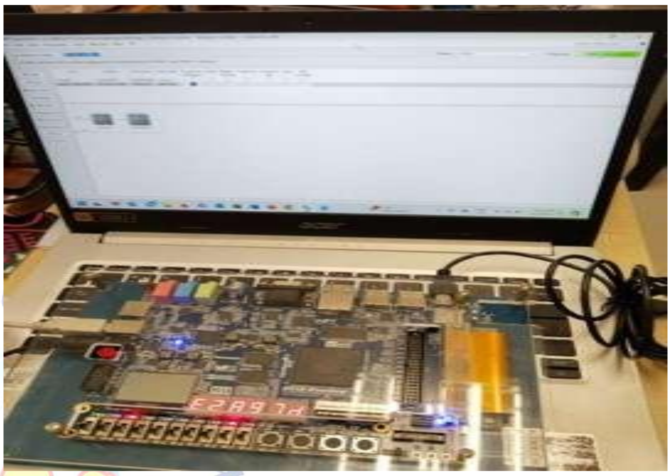


Fig. 7. The laptop's connection to the FPGA board

Image Processing using ModelSim Altera :

To confirm the functionality of the Verilog coding used for digital image processing, functional simulation is carried out in ModelSim. The output waveforms for the design and testbench for the simulation are shown in Figure 8(a),8(b),8(c) respectively for skull, weather, lenna . The clk signal is initially set to low. At the next rising edge of the clock, the gray_module begins reading the hexadecimal data from the picture file . When the gray_data for every pixel is accessible, the each submodule's output data at the starts the simulation. For additional image processing, the brightness_module threshold_module contrast_module, and invert_module receive the gray_data. There is no lag between the data transmission for the gray_module and other submodules ince the always blocks in the submodules are triggered by the change in gray_data. This guarantees that the gray_data and the brightness_data,threshold_data contrast_data, and invert_data are updated during the same clock cycle.

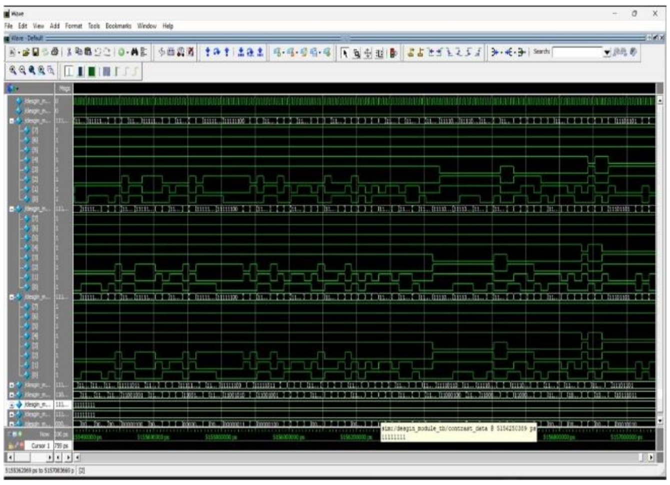


Fig 8(a): output waveforms generated from skull.png

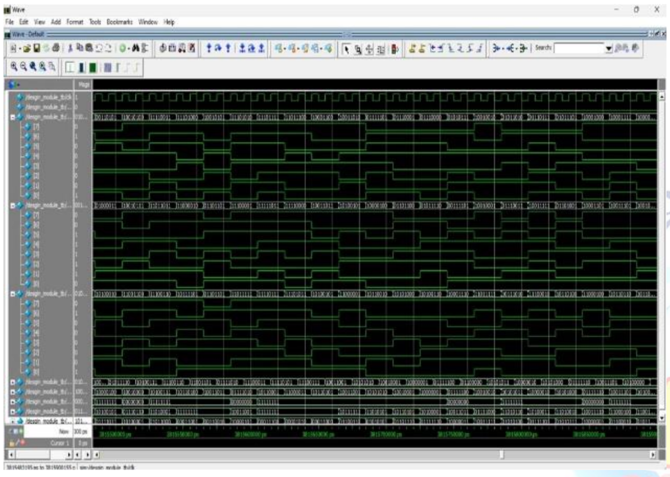


Fig8(b): output waveform generated from weather.png

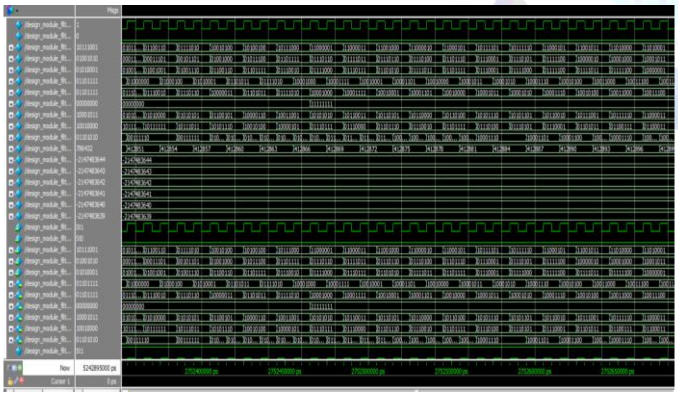


Fig8(c):output waveform from lenna.png

Post-processing using MATLAB :

After the ModelSim simulation is complete, five hexadecimal files are created. The following details are included relevantfiles: threshold_data, contrast_data, invert_data, gray_data, and brightness_data The Top Module block diagram with input hex files and output hex files produced by Verilog HDL code after the simulation process are shown in the figure.

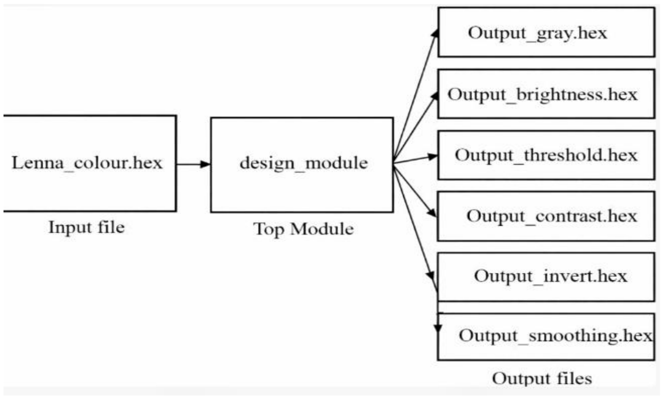


Fig. 9. Block diagram showing the top module's input and output files

At this point, the hexadecimal data that ModelSim produces is transformed into visual images using MATLAB. This stage is essential for visual inspection and validation of the ModelSim simulation's outcomes. six distinct hexadecimal files are used to create a total of six images.

The below figures 10(a),10(b),10(c) shows the five distinct images formed as the outputs of the MATLAB. Fig 10(a) shows "Lenna output images", Fig 10(b) shows "weather output images", Fig 10(c) shows "Skull output images" respectively.

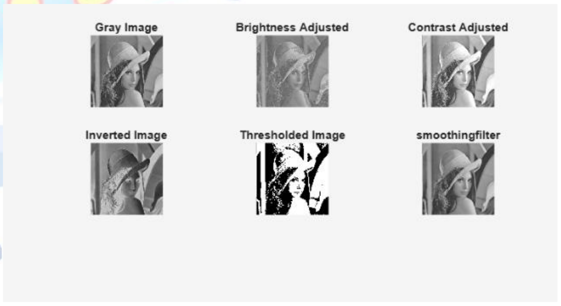


Fig 10(a): conversion of six distinct hexadecimal output formats for the lenna.png file

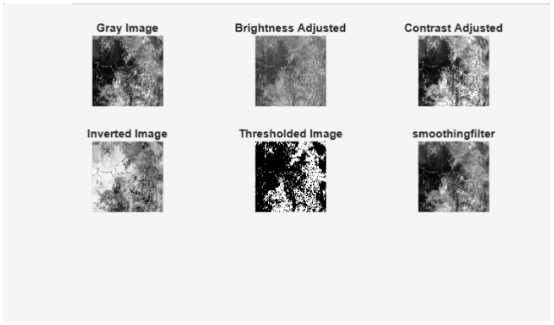


Fig 10(b) : conversion of six distinct hexadecimal output formats for the weather.png file

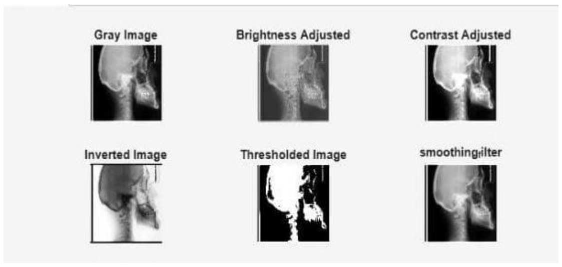


Fig 10 (c): conversion of six distinct hexadecimal output formats for the skull.png

CONCLUSION

Greyscale conversion, brightness modification, thresholding, contrast correction, inversion, smoothing are the digital image processing algorithms used in this project. Both software and hardware are necessary for the application of these techniques in digital image processing. The algorithms are implemented only on FPGA hardware. ModelSim Altera, Intel Quartus II, MATLAB, and Microsoft Paint are used as software. The supplied image is converted from PNG to bitmap format using Microsoft Paint. The bitmap image is converted to hexadecimal representation using MATLAB. The Verilog HDL algorithms are designed using Intel Quartus II and ModelSim Altera. Additionally, they are employed to model and validate image processing processes. The resulting hexadecimal files are once more converted to PNG format using MATLAB, where the pictures are shown so that they can be compared to the input colour picture.

The project's success is determined by how well the digital image algorithms are integrated into the FPGA and implemented in Verilog HDL. The developed algorithms are simulated a verified using Intel Quartus II and ModelSim Altera to ensure the design is functional. Comparisons are done between the input image and the greyscale image, as well as with the other algorithms, to confirm the algorithms' accuracy.

Conflict of interest statement

Authors declare that they do not have any conflict of interest.

References

[1] Moore, Andrew, and Ron Wilson. "FPGAs For Dummies®, 2nd Intel® Special Edition." (2017).

[2] Gonzalez, Rafael C, and Woods, Richard E. "Digital Image Processing, 2nd Ed. Prentice Hall." (2002).

[3] Vanaparthy, Praveen, G. Sahitya, Krishna Sree, and C. D. Naidu. "FPGA implementation of image enhancement algorithms for biomedical image processing." International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering 2, no. 11 (2013): 5747-5753.

[4] Chiuchisan, Iuliana, and Oana Geman. "An approach of fpga technology in skin lesion detection."In 2018 International Conference and Exposition on Electrical And Power Engineering (EPE), pp. 0175-0178. IEEE, 2018. https://doi.org/10.1109/ICEPE.2018.8559866

[5] Zhang, Yunxiang, Xiaokun Yang, Lei Wu, and Jean H. Andrian. "A case study on approximate FPGA design with an open-source image processing platform."In 2019 IEEE Computer Society Annual Symposium on VLSI (ISVLSI), pp. 372-377. IEEE, 2019.https://doi.org/10.1109/ISVLSI.2019.00074

[6] Panappally, J. George Cherian, and M. S. Dhanesh. "Design of graphics processing unit for image processing." In 2014 First International Conference on Computational Systems and Communications (ICCSC), pp. 299-302. IEEE, 2014. https://doi.org/10.1109/COMPSC.2014.7032666

[7] Chaithra, S., Nithya Priya, H.L., Pragna, V., and Spoorthy, M. "Enhancement of Image using Verilog & MATLAB." International Research Journal of Modernization in Engineering Technology and Science (IRJMETS), vol. 5, no. 7, pp. 870–872, 2023, https://doi.org/10.56726/IRJMETS43063

[8] Dhanabal, R., Sarat Kumar Sahoo, V. Bharathi, Kalyan Dowluri, Bh SR Phanindra Varma, and V. Sasiraju. "FPGA based image processing unit usage in coin detection and counting." In 2015 International Conference on Circuits, Power and Computing Technologies [ICCPCT-2015], pp. 1-5. IEEE, 2015. https://doi.org/10.1109/ICCPCT.2015.7159440

[9] Nived, Ch Sai, A. Rohith Kumar, G. Sai Dheeraj, and P. Jithendra. "Image Enhancement based on Verilog Hardware Description Language." International Journal of Engineering Research and Technology. Volume 14, Number 7 (2021), pp. 647-651, 2021.

[10] Azhari, Zul Imran, Samsul Setumin, Emilia Noorsal, and Mohd Hanapiah Abdullah. "Digital image enhancement by brightness and contrast manipulation using Verilog hardware description language." International Journal of Electrical and Computer Engineering 13, no. 2 (2023): 1346.

[11] Chiuchisan, Iuliana. "An approach to the Verilog-based system for medical image enhancement." In 2015 E-Health and Bioengineering Conference (EHB), pp. 1-4. IEEE, 2015. https://doi.org/10.1109/EHB.2015.7391461

[12] "CPU or FPGA for image processing: Which is best?, " Vision Systems Design. Accessed Jul. 04, 2024. [Online].

[13] Raikovich, Tamás, and Béla Fehér. "Application of partial reconfiguration of FPGAs in image processing." In 6th Conference on Ph. D. Research in Microelectronics & Electronics, pp. 1-4. IEEE, 2010.

[14] Keerthi, A., Nikitha, A., Nikhil, A., and Kumar, A. P. "Implementation of Image Enhancement using Verilog HDL". ZKG International, vol. 4, no. 1, pp. 1243– 1263, 2024.

[15] Khudhair, Zaid Nidhal, Ahmed Nidhal Khdiar, Nidhal K. El Abbadi, Farhan Mohamed, Tanzila Saba, Faten S. Alamri, and Amjad Rehman. "Color to grayscale image conversion based on singular value decomposition." Ieee Access 11 (2023): 54629-54638. https://doi.org/10.1109/ACCESS.2023.3279734

[16] "What are the Benefits of using Grayscale Scanning for Specific Applications, such as Medical Imaging?," Electronic Office Systems. Accessed: Jun. 28, 2024. [Online].

[17] "DE10-Standard: Layout, " Terasic Technologies. Accessed: Jan. 14, 2024. [Online].

[18] "Altera Quartus II," University of Nevada, Las Vegas (UNLV). Accessed: Jan. 09, 2024. [Online].

[19] Mentor Graphics Corporation, "ModselSim Tutorial, " 2015. Accessed: Jul. 04, 2024. [Online].

[20] "What is MATLAB?, " MathWorks. Accessed: Jan. 09, 2024. [Online].

[21] Yusefi, Mostafa, Ong Su Yee, and Kamyar Shameli. "Bio-mediated production and characterisation of magnetic nanoparticles using fruit peel extract." Journal of Research in Nanoscience and Nanotechnology 1, no. 1 (2021): 53 61. https://doi.org/10.37934/jrnn.1.1.5361.