International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 11 Issue: 04 | Apr 2024 www.irjet.net p-ISSN: 2395-0072

HDL implementation for real-time image properties adjustments

Sri Lakshmi [1], Mery Lavanya [2], Nikhilesh Mohan [3], Naga Pavan [4], Kavya Reddy [5]

[1]*Associate Professor, Dept. Electronics & Communication Engineering, SR Gudlavalleru Engineering College, Gudlavalleru, Andhra Pradesh, India.*
[2,3,4,5]*Undergraduate Students in, Dept. Electronics & Communication Engineering, SR Gudlavalleru Engineering College, Gudlavalleru, Andhra Pradesh, India.*

Abstract -

Real-time image processing demands high performance that surpasses the capacity of conventional software-based techniques. This paper explores the utilization of Field-Programmable Gate Arrays (FPGAs) with Verilog HDL to achieve hardware-accelerated image enhancement. Verilog implementation for fundamental image processing operations, including thresholding, contrast adjustment, brightness manipulation, and inversion is being implemented. These operations are fulfilled by performing logical operations on the pixels of the subject image. Compared to software-based approaches, this hardware design offers significant speed advantages due to the parallel processing capabilities of FPGAs. This approach is particularly well-suited for real-time applications where immediate image processing is critical.

Key Words: **Image Processing, Verilog HDL, Image Operations**

1. INTRODUCTION

The ever-increasing demand for high-performance digital signal processing (DSP) applications necessitates the search for efficient scheduling methods. Hardware design languages (HDLs) have emerged as a powerful tool for hardware designers, offering a unique blend of simulation capabilities and real-world hardware implementation [1]. This paper explores the benefits of HDLs, particularly their ability to carefully simulate and test digital circuits, also and incorporates important timing considerations how this functionality translates seamlessly into the DSP field, enabling designers to optimize productivity and maintain hardware availability. In addition, the paper explores the use of Field-Programmable Gate Arrays (FPGAs) in conjunction with HDLs. Using conventional HDLs and traditionally configured, FPGAs provide a flexible, cost-effective hardware platform for implementing DSP algorithms [2]. Verilog HDL and Very High-Speed Integrated Circuits (VHSIC) HDL, the two primary HDLs used for FPGA design, will be covered along with their key characteristics and how they help to streamline the DSP design process.

1.1 Image Enhancement Operations

Image enhancement remains a cornerstone of digital image processing, providing significant value in two key application areas. First, it enhances human understanding by improving the interpretation and clarity of visual information in images. Second, it optimizes image data representation for efficient storage, transmission, or tailored use in automated machine perception systems. The basic principle behind any enhancement method is to generate a demonstrably superior result compared to the original image, specifically catering to the requirements of a particular application [3].

There are two main image enhancement methods: spatial domain and frequency domain methods. [4,5] Spatial domain methods directly manipulate the pixels within an image plane, leveraging the image's inherent pixel structure. Frequency domain methods, on the other hand, utilize mathematical transforms to induce enhancements within the image's frequency domain using techniques like the Fourier transform. Some of the earliest and most effective spatial domain techniques involve adjustments to an image's brightness, contrast, or color. These adjustments are often employed to address limitations encountered during image acquisition. For instance, image processing can increase the overall brightness of a target object, revealing previously obscured details, or magnify subtle variations in contrast, allowing for clearer interpretation. As established in various studies, each pixel value is determined solely by its corresponding value at the same position within the image, independent of its neighbors. A function is applied to map the original pixel values to their enhanced counterparts, with functions operating independently of image coordinates being classified as global or homogeneous operations.

1.2 Significance of FPGA

This paper addresses the limitations of software-based image processing, MATLAB especially its struggle to achieve real-time performance due to sequential processing. Field-Programmable Gate Arrays (FPGAs) are used and programmed with Verilog HDL to overcome this constraint and obtain the actual evolution time of the image. [6] An alternative approach I being put forward as FPGAs provide greater efficiency by implementing parallel hardware operations, resulting in significant performance gains compared to traditional software approaches Furthermore, Verilog HDL optimization also provides a potentially cost-effective solution. This research paves the way for significant advances in real-time imaging, opening the doors for applications in critical areas such as medical imaging, autonomous vehicles and security systems.

2. Point Operations

Point functions provide a powerful way to manipulate individual pixel values in an image. [7] This operation modifies the characteristics of individual pixels without affecting the size, form, or local relationships between adjacent pixels of the image as a whole. The new value of each pixel, denoted by a = I (u, v), is determined by its initial value, a = I (u, v), at the same location. This independence from the surrounding pixels allows for precise control of individual intensities. Each pixel in the image (represented by coordinates (u, v)) has a function, f(a), applied to it in order to convert the original values to their improved counterparts.

$$a' \leftarrow f(a)\ I'(u, v) \leftarrow f(I(u, v))$$

Equation 1: Point operations for the proposed approach

From the above Equation 1, f () is a function that represents the coordinates of the image, the operation is classified as global or unitary. Common examples of such homogeneous point operations include optimized intensity conversion ("curve"), image quantization (or "posterization"), global thresholding, gamma correction, and various color conversions

3. Previous Work

Image conversions often introduce quality degradation. To counteract this, image enhancement techniques are employed to improve visual quality. These techniques include contrast stretching, brightness control, inversion, thresholding, and more. While both software and hardware implementations exist for image enhancement, hardware offers superior performance. This project focuses on utilizing a reconfigurable hardware system, specifically Field-Programmable Gate Arrays (FPGAs), to achieve real-time image enhancement. This approach leverages Hardware Description Languages (HDLs) for programming, offering a novel technique within the digital system design domain using Very-Large-Scale Integration (VLSI) [1].

Field-programmable gate arrays (FPGAs) have emerged as a powerful platform for real-time image processing. Compared to programmable digital signal processors (DSPs), FPGAs offer much higher performance due to their application-specific hardware implementation. This work delves into image enhancement algorithms implemented on FPGAs, focusing on techniques such as brightness control, contrast propagation, negative transformation, thresholding, filtering and more. The program uses the System Generator tool in MATLAB to develop a modular image algorithm platform. This platform facilitates the development and implementation of image processing algorithms on FPGAs. After installing the algorithms on the Spartan-3E development board, subsequent experiments are conducted to evaluate the impact on display image quality and resource consumption. This experiment is expected to show the superiority of System Generator for FPGA algorithm design, highlighting its great processing power in real-time image processing tasks [2].

4. Approach

Bitmap image format has been utilized for this process. BMP is a simple, uncompressed format developed by Microsoft. It produces very huge file sizes since it stores every pixel in the image without any compression. BMP can be useful for storing raw image data for further processing or in situations where preserving every detail is crucial.

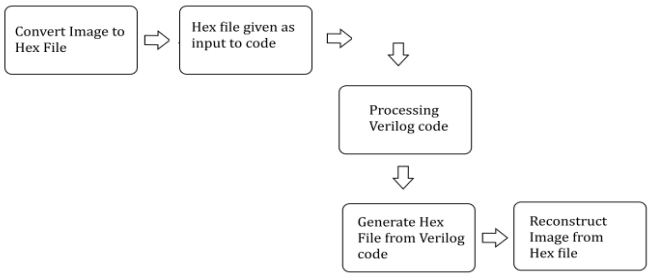

Fig. 1: Block Diagram of the suggested methodology

Although Verilog provides a powerful file format, its primary functionality is limited to ASCII characters. This presents a challenge when working with common models such as BMP, which cannot be directly read by Xilinx software. To overcome this limitation, as an alternative approach to image preprocessing in MATLAB. Here, as shown in the Fig. 1, the BMP file is converted from its original format to a standard hex file containing only the necessary information: RGB vectors for each pixel.

4.1 Image Data

An image's red, green, and blue (RGB) elements are stored in different memory blocks. These memory blocks can be implemented using either external memory chips or internal memory built into the processing unit, depending on factors like cost and processing speed. To access the color information for a specific pixel, the code reads data from a designated memory address for each color channel. The value retrieved from this address corresponds to the intensity of that color component for that particular pixel. The original image is as shown in the Fig. 2.

Fig. 2: Original Image

5. Results

5.1 YUV Conversion

Weighted sum of the red, green, and blue components for each color (red, green, blue) of the output YUV image is performed. These weights are specific to the YUV conversion and determine how each color contributes to the luminance (brightness) and chrominance (color) information. These are then added with offset value 128 and then divided by 256 for scaling. An additional offset (16 or 128) is added to the final values to ensure the output falls within the expected range for YUV components. Fig. 3 represents the YUV converted image.

Fig. 3: YUV converted image

5.2 Image Inversion

Intensity of the pixel is calculated by summing its red, green, and blue components. Assuming that the original color values range from 0 (darkest) to 255 (brightest), the code then inverts each color by subtracting the intensity from the maximum value (255). This basically makes the color detail float on the color, making bright areas appear darker, and vice versa. Finally, the converted colors are stored back into the pixel's data, possibly for use in updating the image. This operation can aid in edge detection. Fig. 4 shows the inverted image for the give original input.

Fig. 4: Inverted Image

5.3 Threshold Operation

Conditional thresholding operation within an image processing framework is implemented by calculating average pixel intensity and comparing it to a predefined threshold. If the average intensity exceeds the threshold, the pixel is set to white (maximum intensity = 255) for all color channels (red, green, blue). Conversely, if the intensity falls below the threshold, the pixel is set to black (minimum intensity = 0). Fig. 5 represents threshold converted image.

Fig. 5: Result for Threshold Operation

5.4 Contrast Adjustment

A pixel's average RGB value is first compared to a predetermined threshold value. Each color channel is brightened by the code if the average intensity is higher than the threshold. To prevent overflow and maintain valid color value, the code checks if the temporary value exceeds the maximum intensity (255). If it does, the color value is set to the maximum value (255). Otherwise, the adjusted intensity from the temporary variable is assigned back to the value. Conversely, if the average intensity falls below the threshold, the code performs a darkening operation by subtracting the same fixed value (Value) from the original intensity of each color channel and storing the result in temporary variables. This could be used for improving image quality and for better analysis. The image after contrast adjustments is shown in Fig. 6.

Fig. 6: Result for Contrast Adjustment

5.5 HSV Conversion

The image's color scheme is now HSV (hue, saturation, value) instead of RGB (red, green, and blue). The RGB values of a pixel are extracted and find the most dominant (highest intensity) and least dominant (lowest intensity) colors. Hue is calculated based on the primary color and the contrast between color components.

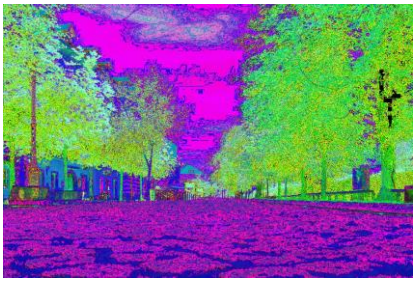

Fig. 7: Result for HSV Conversion

Saturation reflects color purity and is determined by the change in intensity relative to maximum intensity. Finally, value represents the brightness, and is set to the maximum intensity as shown in Fig. 7.

5.6 Brightness Manipulation

Intensity values of each color (red, green, and blue) are retrieved from the memory. This value is then incremented by adding a default constant. The process ensures that the final value does not exceed a maximum limit (255). If the combination pushes the temporary value beyond this limit, it is lowered back to the maximum value to remove invalid character data. Otherwise, the incremented value is used as the final output. Decrementing of the value is performed for brightness degradation with similar steps as enhancement.

Fig. 8: Brightness enhanced Image

Fig. 9: Brightness degraded image

5.7 Gray Level Slicing

Changing pixel intensity based on a specific brightness range is called gray level slicing. The average intensity is calculated by combining its red, green, and blue values. If the average intensity falls within a specific range (based on requirement), the code sets all those color values (red, green, and blue) to their maximum value (typically 255), effectively making the pixel white. Otherwise, the original color values are left as they are. This method selectively brightens pixels within a certain intensity range. For the result which is shown in Fig. 10, the pixels in between 50 and 90 will become 255(white) and remaining will be as it is.

Fig. 10: Result for Gray level slicing

5.8 Grayscale Image

Obtained by calculating weighted sum of the red, green, and blue components. The selection of these weights is intended to mimic the sensitivity of the human eye to various hues. Later divided by 256 to scale the value down to gray scale intensity range. The resulting weighted average represents the grayscale intensity of the pixel. By setting all color channels to the same value, the pixel loses its color information and becomes a shade of gray. This could be

used to create artistic effects as shown in Fig. 11 and modify image for specific purposes.

Fig. 11: Grayscale Image

6. Power Analysis

With the image transferred to memory, the processed data containing RGB components is now saved as a hex file and is converted into a bmp file to acquire the output image. Power analysis in circuit design, especially with Field-Programmable Gate Arrays (FPGAs) from Xilinx, is the process of determining how much power the digital circuit will use in the design stage. Utilizing software that examines the hardware description—which is frequently written in Verilog—and takes into account variables like operation frequency and logic gate usage, this assessment is accomplished. A truly efficient system sets a balance between high performance and low energy usage.

So, the power observed after performing the above operation were obtained as shown in the Fig. 12.

Power Supply Summary			
	Total	Dynamic	Static Power
Supply Power (mW)	183.47	0.00	183.47

Fig. 12: Power Supply Summary

6.1 Device Utilization and Timing Summary:

The percentage of resources, such as logic components and memory blocks, used by the design on the selected device is disclosed in the Device Utilization report. This aids in determining whether the gadget can manage the incorporated functions. Timing Summary analyzes the timing properties, examines flip-flop data setup and hold times, clock frequencies, and signal delays in logic routes. The operations are represented in the below order respectively.

1. Brightness enhancement operation
2. Brightness degradation operation
3. Invert operation
4. Threshold operation
5. Contrast operation
6. Gray level slicing operation
7. HSV conversion operation
8. Gray scale operation
9. YUV conversion operation

Table 1: Timing and Device Utilization Summary

	1	2	3	4	5	6	7	8	9
Slice Logic Utilization									
Number of Slice Registers	0%	0%	0%	0%	0%	0%	0 %	0%	0%
Number of Slice LUTs	0%	0%	0%	0%	0%	0%	3 %	0%	0%
Number used as Logic	0%	0%	0%	0%	0%	0%	3 %	0%	0%
Slice Logic Distribution									
Number of LUT Flip Flop pairs used	127	124	137	113	226	139	6350	94	160
Number with an unused Flip Flop	59 %	58 %	73 %	74 %	76 %	62 %	99 %	70 %	67 %
Number with an unused LUT	7%	8%	7%	8%	4%	7%	0 %	10 %	17 %
Number of fully used LUT-FF pairs	33 %	33 %	18 %	16 %	18 %	30 %	0 %	19 %	15 %
Number of unique control sets	4	4	4	4	7	4	6	4	4
IO Utilization									
Number of IOs	25	25	25	25	25	25	25	25	25
Numbe	4%	4%	4%	4%	4%	4%	4	4%	4%

r of bonded IOBs							%		
Specific Feature Utilization									
Number of Block RAM/FIFO	38 %	38 %	38 %	38 %	38 %	38 %	38 %	38 %	38 %
Number using Block RAM only	288	288	288	288	288	288	288	288	288
Number of BUFG/ BUFGCTRLs	3%	3%	3%	3%	3%	3%	3 %	3%	3%
Number of DSP48E1s							0 %	0%	0%
Timing Summary									
Min Period(ns)	2.186	2.194	4.332	3.319	5.860	3.931	51.582	5.844	7.054
Max frequency (MHz)	457.498	455.830	230.867	301.332	170.657	254.395	19.387	171.130	141.766

7. Conclusion

This project successfully implemented a versatile graphics module using Verilog HDL. The design was simulated and synthesized, demonstrating its functionality. Image enhancement techniques like, YUV transformation, image transformation, HSV transformation, and grayscale transformation are studied and practically performed and the results are observed. By visually comparing original and processed images, the effect of these techniques on image information has been demonstrated.

This work establishes a foundation for future investigations into the broad application of HDL in signal processing modeling. In a future study, additional hardware process simulations will be examined to better evaluate the benefits of this approach. The increasingly robust digital computer aided design (CAD) tools not only offer new development solutions but also open the door to entirely new applications.

REFERENCES

[1] 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

[2] 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.

[3] M. B. Veena, R. Deodurg, V. Shrinidhi and S. Soundarya, "Design of Optimized CNN for Image Processing using Verilog," 2023 4th IEEE Global Conference for Advancement in Technology (GCAT), Bangalore, India, 2023.

[4] Ilham Majid Rabbani, Tito Waluyo Purboyo, "Image Enhancement analysis using various Image Processing Techniques", International Journal of Applied Engineering Research, volume 13, Number 2, 2018

[5] R. C. Gonzalez, R. E. Woods – "Digital Image Processing", Prentice Hall, ISBN 0-13-094659-8, pp. 1-142, 2002.

[6] R. G. Poola, L. P.L and S. S. Yellampalli, "Design of Matlab/Simulink-based Edge Detection operators and hardware implementation on ZYNQ FPGA," 2023 Second International Conference on Electrical, Electronics, Information and Communication Technologies (ICEEICT), Trichirappalli, India, 2023.

[7] Dr. Sagar Patel, Krinesh Patel, Keval Patel and Chaitanya Patel, "Image Enhancement on FPGA using Verilog" published in International Journal of Technical Innovation in Modern Engineering &Science (IJTIMES) Impact Factor: 5.22 (SJIF-2017), e-ISSN: 2455-2585 Volume 5, Issue 03, March-2019

[8] Priyanka S. Chikkali, K. Prabhushetty - "FPGA based Image Edge Detector and Segmentation", International Journal of Advanced Engineering Sciences and Technologies, vol. no. 9, issue no.2, pp187- 192, ISSN 2230-7818, 2011.

[9] Hasnae El Khoukhi, My Abdelouahed Sabri, "Comparative study Between HDLs Simulation and MATLAB for Image Processing" International Conference on intelligent Systems and Computer Vision (ISCV), IEEE, April 2018

[10] Zhaochao Shi, Hui Wu, Weiming Mao, Jing Wang, Chao Zhang, "Implementation of An Automatic Image Enhancement Algorithm for Contrast Stretching on FPGA" Published in 2020 IEEE 9th Joint International

Information Technology and Artificial Intelligence Conference, December 2020

[11] Muhammed Yildirim, Ahmet Cinar, "Simultaneously Realization of Image Enhancement Techniques on Real-Time FPGA", 2019 International Artificial Intelligence and Data Processing Symposium (IDAP), IEEE, September 2019

[12] Avra Ghosh, Sangita Roy, Sheli Sinha Chaudhuri, "Hardware Implementation of Image Dehazing Mechanism using Verilog HDL and Parallel DCP", 2020 IEEE Applied Signal Processing Conference (ASPCON), December 2020

[13] Chaitra, Nithya Priya, Pragna, Spoorthy, "Enhancement of Image using Verilog & MATLAB", International Research Journal of Modernization in Engineering Technology and Science, e-ISSN: 2582-5208, vol.no. 05, issue no.07, July 2023