International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 09 Issue: 01 | Jan - 2025 SJIF Rating: 8.448 ISSN: 2582-3930

Image Processing using VITIS Model Composer

Ms. Aruna Dore[1], K.H.N.M.Ramakrishna[2], L.premanandareddy[3], pritham ashok rudrapur[4]

[1]*Assistant Professor, Electronics and Communication Engineering, Presidency University*

[2]*Student, Electronics and Communication Engineering, Presidency University*

[3]*Student, Electronics and Communication Engineering, Presidency University*

[4]*Student, Electronics and Communication Engineering, Presidency University*

Abstract -

This study outlines a method for image processing utilizing the Vitis Model Composer. AMD's Vitis Model Composer includes comprehensive libraries that support a variety of algorithms. In this work, it is integrated within the MATLAB Simulink environment. A model-based design approach is employed to implement various image processing algorithms. To ensure accuracy, hardware co-simulation is conducted for result verification. The study covers several image processing algorithms, including those for converting RGB images to grayscale, creating image negatives, enhancing images, subtracting backgrounds, applying thresholding, and performing morphological operations such as erosion, dilation, and masking, using AMD's provided blocks.

Key Words: optics, photonics, light, lasers, templates, journals

1.INTRODUCTION

Growing demand for efficient and scalable systems for image processing. This project focuses on the software-based implementation of image processing algorithms using AMD Vitis Model Composer, a robust platform integ The rapid advancement in digital imaging and real-time processing technologies has led to a rated with MATLAB Simulink. The project aims to model and validate core image processing algorithms in a simulated environment, without transitioning to hardware implementation. The use of Vitis Model Composer provides a modern, high-level abstraction for designing, testing, and optimizing algorithms for FPGA applications, even in a software-only workflow.In this process, an image serves as the input, while the output may consist of key parameters, characteristic features, or an improved version of the image itself. Digital images convey critical information, including object boundaries, orientation, dimensions, and color. To accurately determine an object's shape and structural attributes, identifying its edges is a fundamental step. Additionally, this paper outlines the architecture of an image processing application Using Vitis Model Composer, an extension of MATLAB-Simulink. The tool offers a specialized Block library, "Blocks AMD," which includes mapped entities, ports, signals, architectures, and attributes. These elements enable the generation of synthesis for FPGAs, HDL simulations, and development tools. When converted into VHDL, the design retains the hierarchical structure of the Simulink model. The final implementation of the design on an FPGA is achieved using Vitis Model Composer,demonstrating its effectiveness in real-time image processing.

1.1 AMD Vitis Model Composer

AMD Vitis Model Composer is an advanced design tool tailored for system developers to create, simulate,and deploy signal processing, image processing, and machine learning algorithms on FPGAs and adaptive SoCs. Seamlessly integrated with MATLAB and Simulink, it offers a streamlined workflow for converting high-level algorithmic models into hardware-ready implementations using a model-based design approach.AMD Vitis Model Composer is a powerful tool for designing and implementing high performance FPGA-based systems. Its integration with MATLAB and Simulink allows engineers tostreamline their workflow, from algorithm development to hardware deployment. With features like model-based design, automatic code generation, and hardware-in-the-loop testing, it is an ideal choicefor applications in image processing, signal processing, and machine learning. By simplifying complex design tasks and optimizing for FPGA resources, it helps accelerate development and ensures scalable,efficient solutions.

1.2 Design Flow for Image Processing Using AMD Vitis Model Composer

The process of designing image processing systems with AMD Vitis Model Composer starts by analyzing the project requirements and identifying key tasks like grayscale conversion and image enhancement. The next step involves building the model in MATLAB Simulink, utilizing Vitis Model Composer blocks. Once the model is ready, simulations are performed to confirm its functionality. Hardware co-simulation is then conducted to test the design's real-time performance on an FPGA. After ensuring the accuracy of the results, the model is fine-tuned for optimal resource usage and performance. Subsequently, HDL code is generated and synthesized using tools such as Vivado, leading to comprehensive hardware testing. The final phase involves documenting the entire process and deploying the FPGA-based image processing solution.

will be built Taking aid from the framework which is being discussed in this section alongside its key components.The key components to the framework include: Image Viewer Unit, Image Pre Processing Unit, Image Source Unit, Post Image Processing Unit, and finally the Hardware Algorithm Unit.

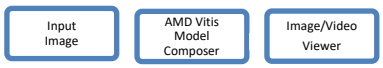

Fig – 1 : General Flow of Image Processing

2 Interfacing with AMD Vitis Model Composer

The Simulink environment treats each number in a simulation as a double which is essentially a real number and In Simulink a double is fairly represented as a 64 bit 2's complement floating point number. Unfortunately, this number system consumes a lot of resources, which is inefficient on FPGAs.

Thus, the AMD blocksets make use of n-bit fixedpoint numbers n, which means that there has to be a translation when AMD- blocks interacts with Simulink blocks .Gateway In, Gateway Out, and Sampling are utilized during this transition

3. Design Framework for Image Processing in AMD Vitis Model Composer

With the pipeline being designed in a modular fashion it can then be deployed on AMD FPGA with relative.

While using the AMD Vortis model composer the Image Processing pipeline will be built Taking aid from the framework which is being discussed in this section alongside its key components.The key components to the framework include: Image Viewer Unit, Image Pre Processing Unit, Image Source Unit, Post Image Processing Unit, and finally the Hardware Algorithm Unit. With the pipeline being designed in a modular fashion it can then be deployed on AMD FPGA with relative.

Fig -2 : Design flow of implementation of image processing

weight, followed by red and blue.

4 . Image Pre-Processing Unit

There are tasks that take place beforedata is sent out from an FPGA. The location where These tasks take place is known as image pre-processing unit and its basic aim is to prepare data prior to performing any algorithms. It is also part of image processing and helps in adjusting images otherwise making them fit for the Software.

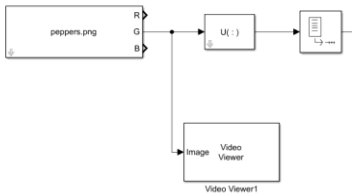

Fig - 3 : Image Pre-Processing Unit

5 . Image Post-Processing Unit

The Image Post-Processing Unit is a critical stage in the image processing pipeline implemented in AMD Vitis Model Composer. Its primary purpose is to transform the processed data, typically in a one-dimensional (1-D) array or pixel stream format, back into a two-dimensional (2-D) image matrix that can be displayed, analyzed, or stored. This stage ensures that the processed image can be visualized correctly, providing insight into the effectiveness of the preceding hardware algorithms.The post-processing unit comprises four main blocks: Data Type Conversion, Buffer, 1-D to 2-D Conversion, and Video Viewer. Each block plays a vital role in ensuring smooth conversion and reconstruction of the processed image for display.

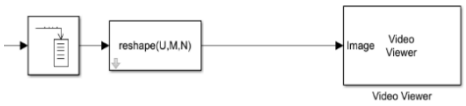

Fig-4: Image Post-Processing Unit

6. Converting RGB Image to Grayscale with AMD Vitis Model Composer

The conversion of RGB images to grayscale is a critical step in many image processing applications.

It reduces computational complexity by transforming a three-channel image into a single-channelrepresentation. AMD Vitis Model Composer provides a robust and efficient platform to implement this process on FPGA hardware, leveraging its integration with MATLAB and Simulink.

6.1 Grayscale Conversion Formula

An RGB image consists of three separate color components: Red (R), Green (G), and Blue (B). The grayscale intensity for each pixel is calculated as a weighted sum of these channels:This formula accounts for the human eye's varying sensitivity to different wavelengths, giving the green channel the highest become light. This technique is used in various applications like image enhancement, feature extraction, and artistic transformations.

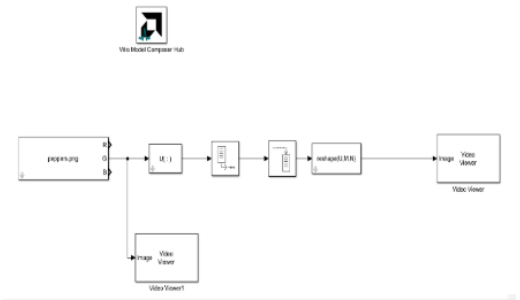

Fig - 5 : Algorithm for RGB Image to Grayscale Image

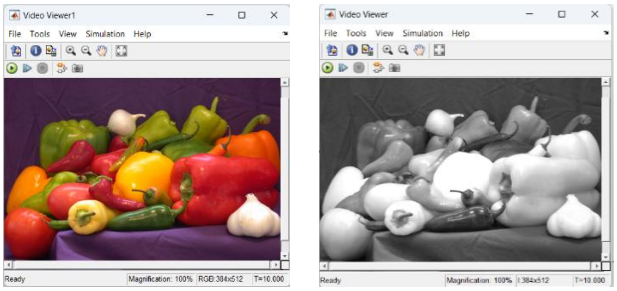

Fig - 6 : Output for RGB Image to

GrayscaleImage

7 . Grayscale Image Negative

Creating a negative of a grayscale image involves inverting the pixel intensity values,

7.1. Concept of Image Negation

The negative of a grayscale image is created by subtracting the intensity of eachpixel from the highest possible value.. For an 8-bit grayscale image, where pixel intensity ranges from 0 to 255, the negative image is computed a I(negative)=255−I(original)

Here, *I*(original) denotes the original intensity of the pixel,and *I*(negative) is the pixel value in the resulting negative Image.

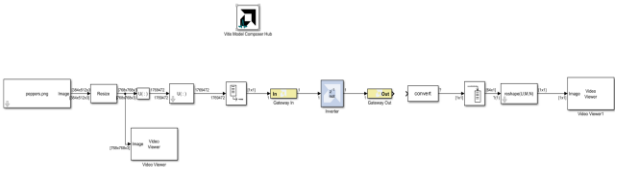

Fig – 7 : Algorithm for Image Negative for grayscale

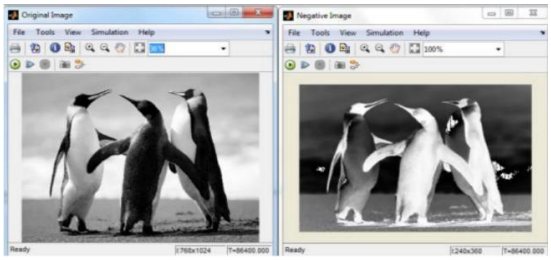

Fig – 8 : Output for Image Negative for grayscale

8 . Image Thresholding Using AMD Vitis Model Composer

8.1 Overview of Image Thresholding

Image thresholding is a fundamental technique in image processing used to simplify a grayscale or color image into a binary representation. By comparing pixel intensity values to a predefined threshold, each pixel is classified as either black (below the threshold) or white (above the threshold). This process reduces an image to two intensity levels, making it easier to isolate and analyze objects of interest.

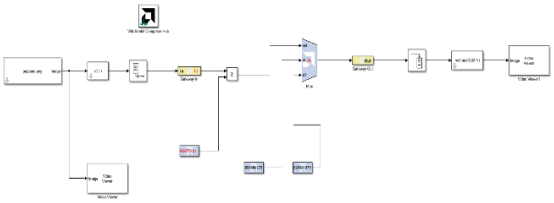

Fig-9: Algorithm for Image Thresholding

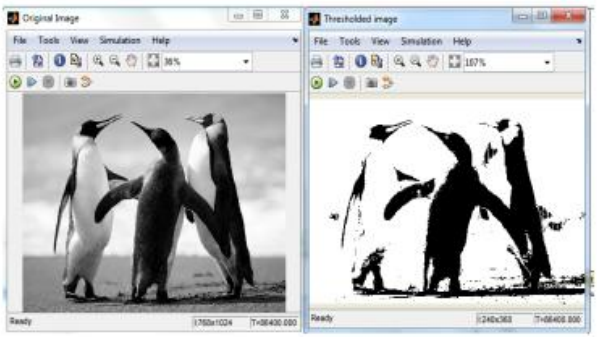

Fig-10 : Output Image Thresholding

9 . CONCLUSION

The project showcases the effective implementation of crucial image processing techniques, such as grayscale conversion, image negativity, and thresholding, using AMD Vitis Model Composer on FPGA.

- **Grayscale Conversion**: This method simplifies color images into grayscale, reducing data complexity and making subsequent processing more efficient.
- **Image Negativity**: By inverting the pixel values, this

technique enhances image features, particularly useful in low-contrast settings.

- **Thresholding**: This process divides images based on intensity, aiding in real-time applications like object detection and feature extraction.

Leveraging the parallel processing capabilities and speed of FPGA, the project demonstrates its suitability for real-time image processing needs. The accuracy and efficiency of these algorithms have been confirmed through hardware co-simulation, underscoring their reliability for real-world use.

REFERENCES

1. DSP System Generator User guide release 10.1, March 2008**.**

2. Xilinx System Generator, User's Guide, www.xilinx.com

3. White paper: Using System Generator for Systematic HDL Design, Verification and

4. Validation.WP283 (v1.0) January 17, 2008.

5. R.G.R. Woods, “Digital Image Processing”, New Jersey, Prentice-Hall, 2008

6. Sofia Mosesson. Using MATLAB® and Simulink® for Image and Video Processing, Application Engineer; 2007.

7.] Zhang, Li Tao, Ming-Jung Seow and Vijayan K. Asari Ming Z., "Degign of Efficeient Flexible Architecture for Color Image Enhancement," Lecture Notes in Computer Science, Advances in Computer System Architecture Vol.4186, PP.323-336, 2006.

8. S. Hasan, A. Yakovlev, and S. Boussakta, "Performance efficient FPGA implementation of parallel 2-D MRI image filtering algorithms using Xilinx system generator" IEEE International Conference on Communication Systems Networks and Digital Signal Processing (CSNDSP), pp. 765-769, July 2010.

9. Fatemeh Taherian, Davud Asemani, Elham Kermani, "Design and Implementation of Edge Detection and Contrast Enhancement Algorithms Using Pulse-Domain Techniques", The Fourth International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies, pp.495-499, 201 O

10. Matthew Own by, Dr Wagdy.H. mhmoud, “A design methodology for implementing DSP with xilinx system generator for Matlab”, processdings of 35th south eastern symposium, Vol 15, page 2226-2238, 2006.

11. Alain Merigot, “Revisiting image splitting”, Proc of 12th international conference on image analysis and processing, page 314-319, 2003

12. “Edge-forming methods for image zooming,” by Y. Cha and S. Kim J.Math. Imag. Vis., vol. 25, no. 3, pp. 353– 364, 2006.