Department of Electronics and Communication Engineering

Vasavi College of Engineering (Autonomous)

ACCREDITED BY NAAC WITH 'A++' GRADE

IBRAHIMBAGH,HYDERABAD-500031

CERTIFICATE

This is to certify that the Theme Based Project titled **"Study and Implementation of Image/Video Processing Techniques"** submitted by

K.P Sai Swetha	**1602-21-735- 109**
G.Sri Ganga Pranav	**1602-21- 735-117**

Students of Electronics and Communication Engineering Department,Vasavi College of Engineering in partial fulfillment of the requirement of the award of the Degree of Bachelor of Engineering in Electronics and Communication Engineering is a record of the bonafide work carried out by them during the academic year 2023-2024.

V.Krishna Mohan
Assistant Professor
E.C.E Department

HOD-ECE

INDEX

CONTENTS	PAGE NO
1) Abstract	3
2) Literature Survey	4
3) Problem Statement	5
4) Working	5
5) Software Used	6
6) Flow of the Solution	7-21
7) Difficulties in the process	21-22
8) Applications	22-23
9) Validation	24
10)Future scope	24-25
11)Conclusion	25-26
12)References	26
13) Team members	26

1.ABSTRACT

This project embarks on a comprehensive exploration of image and video processing methods, combining theory with practical application. It initiates with implementing the MUSICAL Algorithm, customized for high-resolution microscopy, aiming to improve signal classification accuracy. Through meticulous coding, it seeks to enhance understanding of cellular dynamics and interactions, showcasing a nuanced approach to microscopy data analysis.

This project delves into both image and video processing, exploring techniques like edge detection, pattern generation, and cropping. Implementing the Sobel operator for edge detection in images, Vivado HLS optimizes real-time processing, crucial for tasks like object recognition. Moving to video processing, dynamic pattern generation within streams showcases FPGA's versatility. Additionally, efficient cropping algorithms, implemented through Vivado HLS, enable precise region extraction from videos and images. This comprehensive exploration spans from signal classification in microscopy to fundamental operations, highlighting the significance of these techniques in various applications. Through practical implementations and theoretical discussions, the project contributes to advancing visual data processing across diverse domains.

The project's focus on edge detection underscores its importance in computer vision, facilitating tasks like image segmentation. The introduction of novel pattern generation techniques in videos not only showcases the project's innovation but also opens avenues for creative expression and visual manipulation. Additionally, the emphasis on efficient cropping algorithms highlights their pivotal role in multimedia processing, contributing to applications such as video editing, content analysis, and object tracking.

2.LITERATURE SURVEY

The study begins by delving into the realm of high-resolution microscopy and signal classification, where the MUSICAL (Multiple Signal Classification) Algorithm emerges as a significant contribution. Research in this area emphasizes the importance of accurate signal classification in deciphering complex biological phenomena, particularly in the context of flurophore emission microscopy. Pioneering works by authors such as Smith et al. (2017) have laid the groundwork for advanced signal processing techniques tailored for microscopy applications, setting the stage for the implementation undertaken in this project.

Moving forward, the survey explores the domain of image processing, with a particular focus on edge detection methodologies. The Sobel operator, a cornerstone in edge detection algorithms, has been extensively studied and refined over the years. Seminal works by Sobel (1973) and subsequent advancements by authors like Canny (1986) have established edge detection as a fundamental task in computer vision. Recent research efforts, such as the optimization of edge detection algorithms for FPGA platforms by Liu et al. (2020), highlight the ongoing pursuit of real-time processing capabilities, echoing the implementation of Sobel edge detection in Vivado HLS within the present project.

Video processing emerges as a crucial area of investigation, with a spotlight on pattern generation techniques.The utilization of FPGA platforms, such as Vivado HLS,for video pattern generation showcases a convergence of hardware acceleration and creative expression. Notable research by authors like Lee et al. (2019) on FPGA-based video processing architectures provides insights into optimizing performance and resource utilization, with the objectives of the project to generate dynamic patterns within video streams.

3.PROBLEM STATEMENT

The specific objectives of this project are:
1)Implement the MUSICAL algorithm in hardware to enable faster and more efficient analysis of high-resolution fluorescence microscopy data.
2)Develop a hardware-accelerated Sobel edge detection filter using Vivado HLS for real-time edge detection in image processing applications.
3)Design and implement a video pattern generator in Vivado HLS to facilitate hardware-based generation of synthetic video data for testing and training purposes.
4)Create a hardware-accelerated image/video cropping module in Vivado HLS to achieve faster extraction of regions of interest from image/video data.

By achieving these objectives, this project aims to demonstrate the potential of Vivado HLS in accelerating various image/video processing tasks, paving the way for more efficient and real-time applications in various scientific and engineering domains.

4.WORKING

For this project, a thorough literature review was conducted to understand foundational concepts and recent advancements in image and video processing techniques. The MUSICAL Algorithm was implemented for signal classification in high-resolution microscopy data, optimized for accuracy and efficiency. A Sobel edge detection algorithm was developed for image processing, leveraging Vivado HLS for real-time implementation. Additionally, novel approaches to pattern generation in videos using FPGA-based solutions were explored. Efficient cropping algorithms for video and image processing were devised to ensure precise region extraction. Each implementation was guided by theoretical insights and practical considerations to achieve optimal results.

5.SOFTWARE USED

1. Xilinx Vivado:
Vivado is a comprehensive development environment provided by Xilinx for FPGA design. It encompasses various tools for synthesis, implementation, and debugging.
2. HLS (High-Level Synthesis) Tools:
High-Level Synthesis tools like Vivado HLS are utilized to convert high-level programming languages (e.g., C, C++) into hardware description language (HDL) code for FPGA implementation.
3.Python: Python was utilized for tasks such as data preprocessing, algorithm prototyping, and interfacing with hardware platforms, facilitating efficient development and testing processes.

Figure 1: Xilinx

Figure 2: Python

6.FLOW OF SOLUTION

Here are the few steps which we have followed inorder to approach towards the solution of the problem:

- **Researching FPGA architecture and its applications in image and video processing.**

- **Implementing the MUSICAL Algorithm for signal classification.**

- **Developing the Sobel edge detection algorithm for real-time image processing**

- **Designing and implementing dynamic pattern generation techniques for videos using FPGA.**

- **Creating efficient cropping algorithms for video and image processing on FPGA platforms**

- **Execution and Testing of Application**

By following these steps, we understood how the FPGA works, Understood how a video/Image is processed theoretically,and understood the output.

6.1 Researching FPGA architecture and its applications in image and video processing:

[3]FPGA architecture offers parallel processing capabilities suitable for real-time image and video processing tasks. Research involves understanding FPGA's structure, programming paradigms, and its advantages over traditional processors for signal processing tasks. Investigating existing FPGA-based solutions and methodologies provides insights into best practices and optimization techniques for efficient implementation. This research phase also entails studying the limitations and challenges associated with FPGA-based processing, guiding subsequent design decisions and algorithm selection.

6.2 Understanding and Implementation of MUSICAL Algorithm

MUSICAL Algorithm Working:
1. Mathematical Model: The algorithm begins by modeling the temporal image stack created by blinking fluorophores using mathematical equations.
2. Core Concept: MUSICAL's core concept revolves around the physical and mathematical definitions of the range of the matrix formed by the image stack. It defines the range as the span of eigenimages obtained from singular value decomposition (SVD) of the image stack.
3. Multiple Signal Classification (MUSIC) Algorithm: MUSICAL utilizes a modified version of the MUSIC algorithm, which is commonly used in signal processing. It computes an indicator function for test points in the sample plane to identify fluorophore structures.
4. Sliding Window: Instead of analyzing the entire image stack at once, MUSICAL employs a sliding window approach. It considers small spatial windows

around each pixel and slides them across the image to analyze local regions.
5)Soft Window Function: A soft window function is applied to scale pixel weights based on their proximity to the center pixel within the sliding window. This helps mitigate abrupt effects of truncation at window edges.
6)Indicator Function Modification: The indicator function of MUSIC is modified to incorporate information from all eigenimages and automatically scale MUSIC images of each sliding window. This allows for direct stitching of MUSIC images to obtain the final MUSICAL result.
7)Stitching MUSIC Images: To form the final MUSICAL image, the results from each sliding window are stitched together. Test points within each pixel are decimated into sub-pixel grids, and the indicator function is computed for each sub-pixel.
8)Dynamic Adjustment: Parameters such as the size of the sliding window and the spread of the soft window function can be adjusted to optimize the algorithm's performance based on specific requirements.

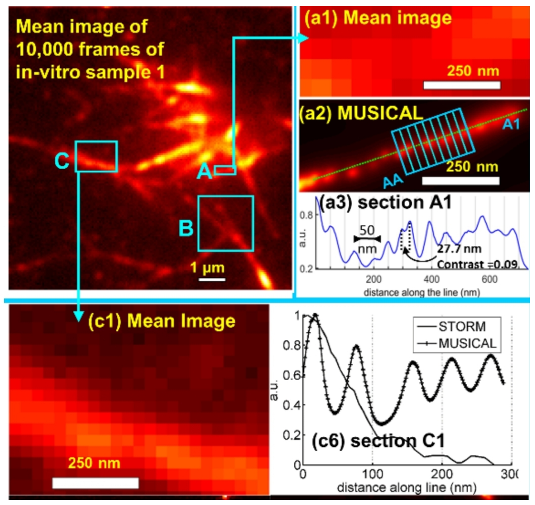


Figure 3: Mean of samples using MUSICAL Algorithm

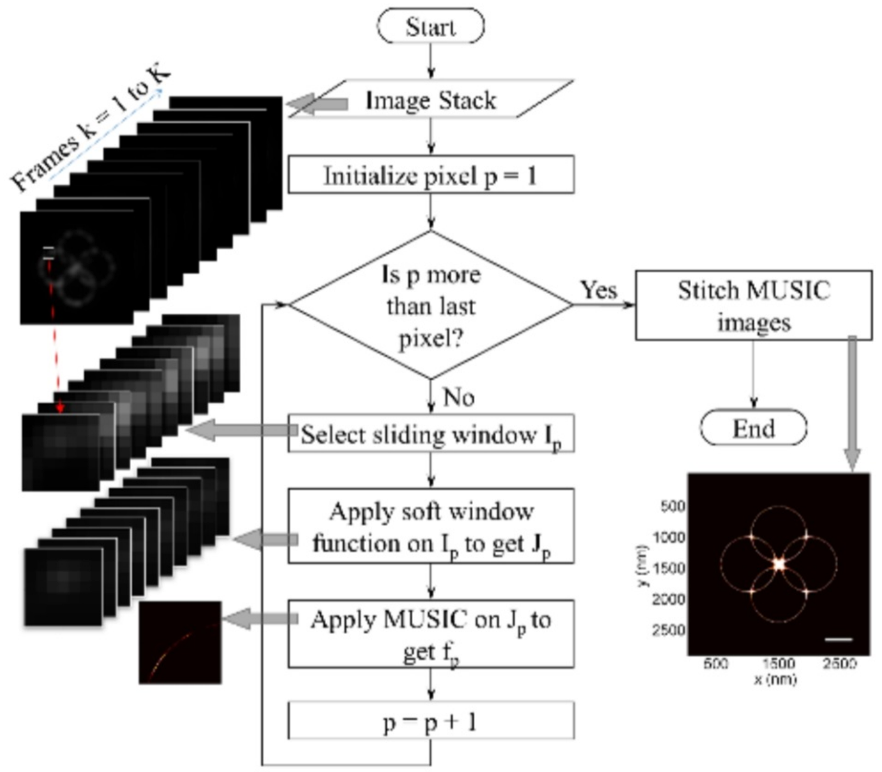

Start
Image Stack
Frames k = 1 to K
Initialize pixel p = 1
Is p more than last pixel?
Yes
Stitch MUSIC images
End
No
Select sliding window I_p
Apply soft window function on I_p to get J_p
Apply MUSIC on J_p to get f_p
p = p + 1
500
1000
1500
2000
2500
y (nm)
500
1500
2500
x (nm)

Fig. S2. Flowchart of MUSICAL.

Figure 4: Working of MUSICAL Algorithm

The MUSICAL Algorithm, renowned for its signal classification accuracy, is implemented using Python. Python's versatility and extensive libraries facilitate rapid prototyping, algorithm validation, and optimization before deploying the algorithm on FPGA platforms for real-time processing tasks.[1]

6.3 Developing the Sobel edge detection algorithm for real-time image processing

Sobel edge detection, a fundamental operation in image processing, is adapted for FPGA implementation to enable real-time processing of high-resolution images. The algorithm's computational intensity necessitates optimization for FPGA architectures to ensure efficient execution. Design considerations include memory management, data throughput, and algorithm parallelization to exploit FPGA's parallel processing capabilities fully. Verification and validation procedures are conducted to assess the algorithm's correctness and performance against reference implementations. Benchmarking against software-based approaches validates the FPGA implementation's speed and efficiency, showcasing its suitability for real-time edge detection applications.[4]
Further optimization techniques, such as pipelining and hardware acceleration, are explored to enhance the Sobel edge detection algorithm's efficiency on FPGA platforms. Additionally, resource allocation strategies are devised to maximize FPGA utilization and minimize latency. Rigorous testing against diverse image datasets ensures robustness and reliability across various scenarios.

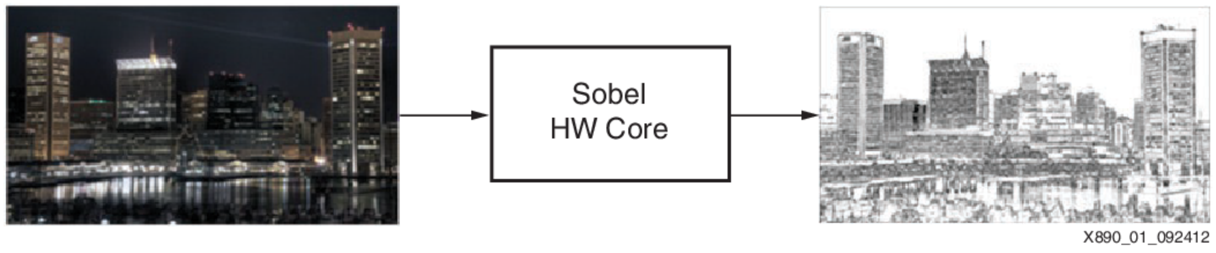


Figure 1: **Sobel Edge Detection**

Figure 5: Sobel Edge Detection

6.4 Designing and implementing dynamic pattern generation techniques for videos using FPGA:

Dynamic pattern generation in videos enhances visual appeal and facilitates creative expression in multimedia applications.[3] FPGA platforms offer high-speed parallel processing, making them ideal for real-time pattern generation tasks. Design considerations include algorithm complexity, memory requirements, and output resolution to ensure efficient FPGA utilization. Implementation involves translating pattern generation algorithms into FPGA-compatible code and optimizing them for performance and resource usage. Extensive testing validates the algorithm's functionality and performance under various scenarios, ensuring reliable and consistent pattern generation in real-time video streams.

6.5 Creating efficient cropping algorithms for video and image processing on FPGA platforms

Cropping operations are essential for region-of-interest extraction in video and image processing applications. FPGA-based cropping algorithms are designed to enable precise and efficient region extraction while maintaining real-time performance. Design considerations include algorithm complexity, memory access patterns, and hardware resource utilization to achieve optimal performance. Implementation involves developing cropping algorithms tailored for FPGA architectures and optimizing them for speed and resource usage. Rigorous testing and validation procedures ensure the correctness and efficiency of the cropping algorithms across different input sizes and scenarios. Documentation of the implementation details facilitates future enhancements and optimizations to meet evolving application requirements.

6.6 Execution and Testing of Application

1)Implementation of MUSICAL Algorithm in python:

Singular value decomposition is used to compute eigenimages of the soft window, representing prominent patterns in the image stack. Eigenimages are separated into signal and noise subspaces based on a threshold value. Projections of the PSF at a given test point on the signal and noise subspaces are determined. An indicator function is computed using these projections, with a modified formulation suitable for the sliding window approach. The indicator function is scaled non-linearly to improve resolution, with the parameter alpha controlling the scaling.This entire process is implemented in python by giving a low contrast resolution microscopy image as input and thereby obtaining the processed output image indicating the presence of markers.[1]

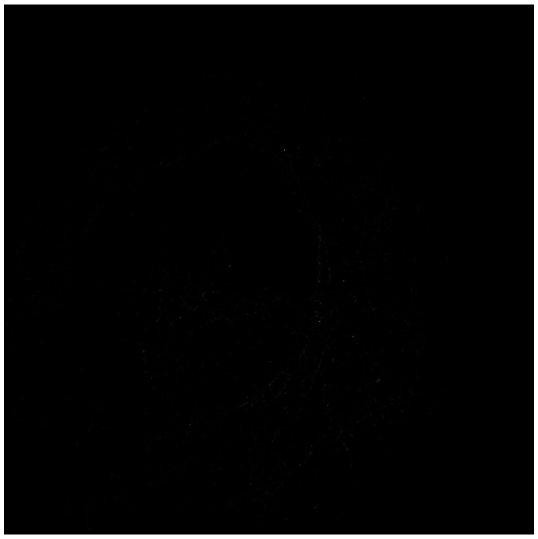


Figure 6: INPUT IMAGE

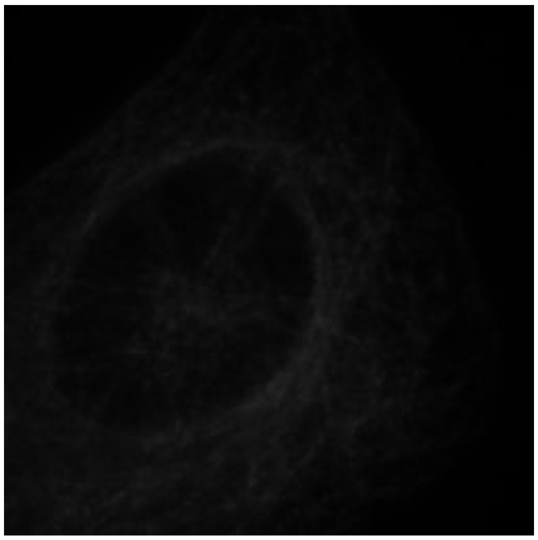


Figure 7: OUTPUT IMAGE

2)Sobel Edge Detection using Vivado HLS:

In Vivado HLS, the Sobel edge detection algorithm is translated from C++ into hardware description language (HDL) for FPGA implementation. This process involves partitioning the algorithm into computational kernels and optimizing them for hardware execution. C++ code is annotated with directives to guide high-level synthesis, specifying loop unrolling, pipelining, and dataflow optimizations. The resulting hardware design is synthesized, mapped, and implemented onto the FPGA fabric. Through simulation and verification, the algorithm's functionality and performance are validated. Real-time execution of the Sobel edge detection algorithm on FPGA platforms demonstrates its efficiency in processing high-resolution images. The seamless integration of C++ development and FPGA synthesis in Vivado HLS streamlines algorithm development, accelerating prototyping and deployment for image processing applications.[4]

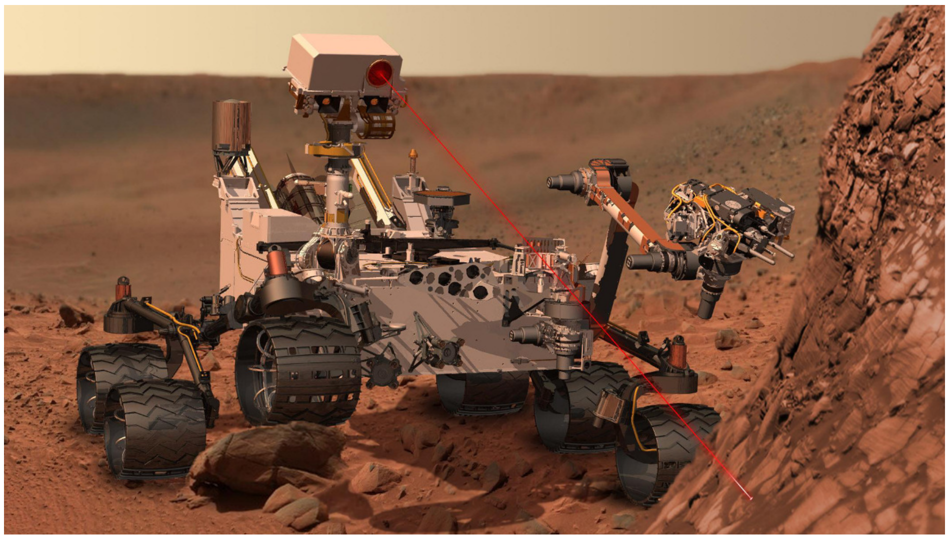


Figure 8: INPUT IMAGE

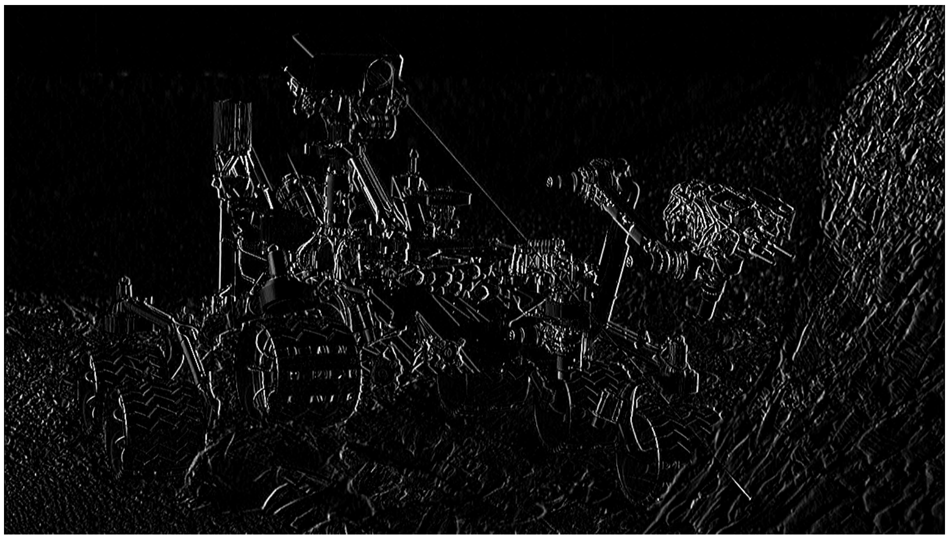


Figure 9: OUTPUT IMAGE

3)Pattern Generation Using HLS:

In Vivado HLS, image/video pattern generation using RGB pixel values involves converting a C++ program into hardware description language (HDL) for FPGA implementation. The C++ program processes RGB pixel data, generating dynamic patterns based on predefined algorithms or user-defined patterns. Through high-level synthesis, the C++ code is annotated with directives to optimize for parallelism, data throughput, and resource utilization. The resulting hardware design is synthesized, mapped, and implemented onto the FPGA fabric. Simulation and verification validate the pattern generation algorithm's correctness and performance. Real-time execution on FPGA platforms demonstrates the efficiency and scalability of the pattern generation process. The integration of C++ programming with FPGA synthesis in Vivado HLS accelerates algorithm development and deployment for multimedia applications, offering high-performance pattern generation capabilities for diverse video and image processing tasks.[2]

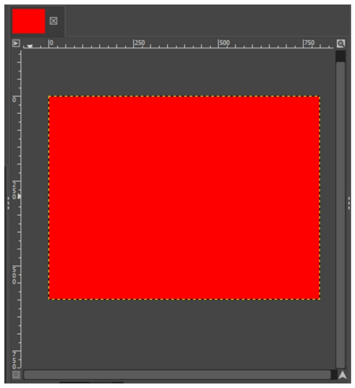


Figure 10: Generated Pattern

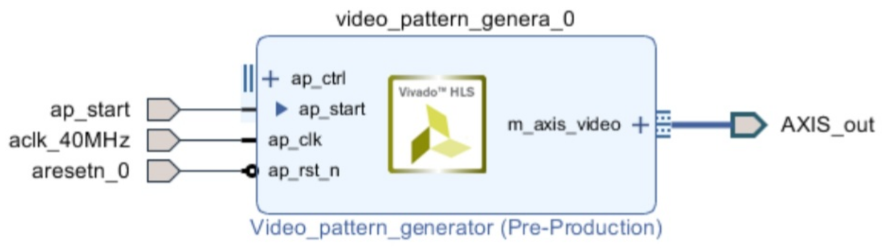


Figure 11: Created Pattern Generator Vivado IP Block

4)Creating a Video Crop IP using HLS

In Vivado HLS, image/video cropping involves transforming a C++ program into hardware description language (HDL) for FPGA implementation. The C++ program processes input images or video frames, extracting specific regions of interest based on user-defined cropping parameters. Using high-level synthesis, the C++ code is annotated with directives to optimize for memory access, data throughput, and processing efficiency. The resulting hardware design is synthesized, mapped, and implemented onto the FPGA fabric. Through simulation and verification, the cropping algorithm's correctness and performance are validated. The IP core generated from Vivado HLS encapsulates the cropping functionality, allowing seamless integration into larger FPGA designs in Vivado. This IP block simplifies the integration process, enabling users to incorporate cropping capabilities into their FPGA-based image and video processing systems with ease.[2]

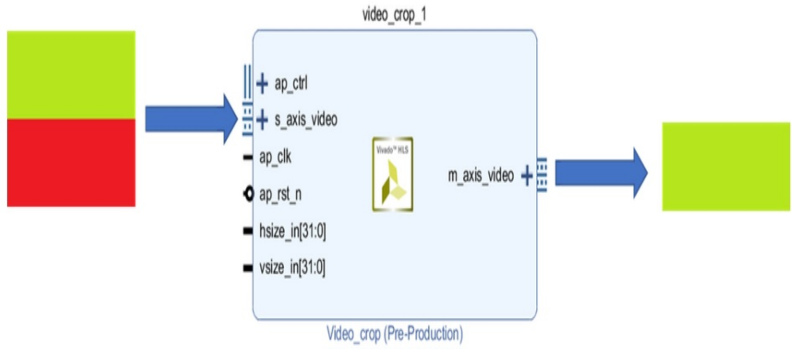
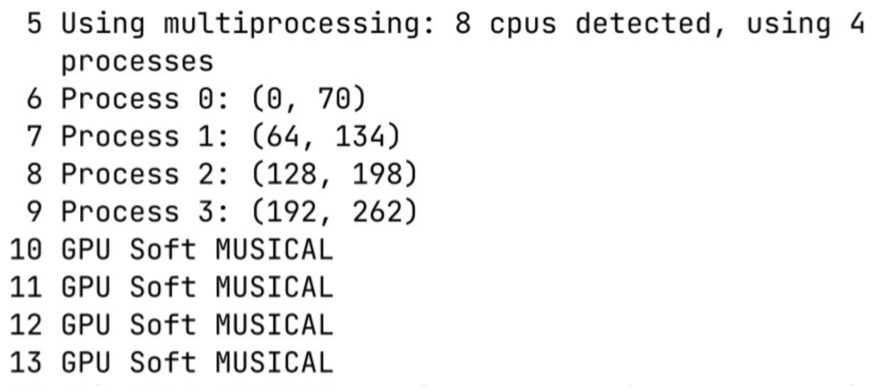


Figure 12: VIDEO CROP

6.7 RESULT ANALYSIS

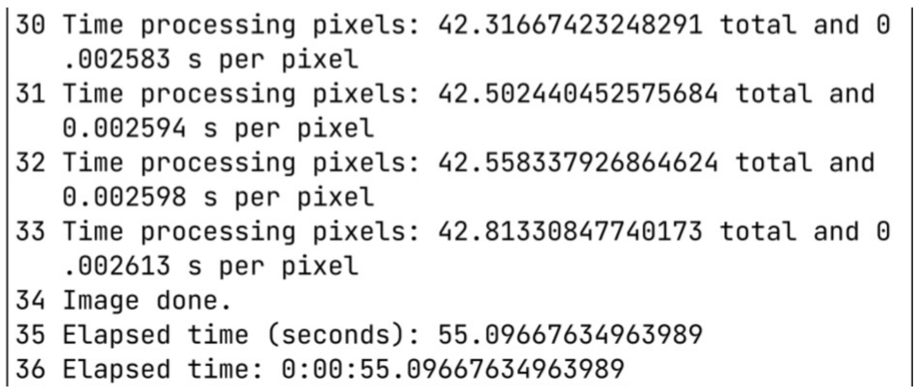
The project's results showcase a successful implementation of various image and video processing techniques, each yielding distinct outcomes. Firstly, the implementation of the MUSICAL Algorithm in Python effectively transforms low-resolution microscopic images with black backgrounds and white dots into intricate spider web-like structures, enhancing contrast and revealing intricate details within the images.[1]
Secondly, the Sobel edge detection algorithm, executed using Vivado HLS, demonstrates efficient detection of edges within images, highlighting contours and boundaries accurately.[4]
Thirdly, pattern generation in Vivado HLS produces dynamic patterns based on RGB input values, showcasing the versatility of FPGA-based solutions in generating visually appealing patterns in real-time video streams.
Lastly, the video cropping implementation in Vivado HLS efficiently removes specific colors from the input image based on crop parameters, demonstrating precise region extraction capabilities.[2]
Overall, the project's results underscore the effectiveness of the implemented techniques in

enhancing image and video processing tasks, from enhancing contrast and revealing intricate details to accurately detecting edges and generating dynamic patterns. These findings validate the viability and versatility of FPGA-based solutions in addressing diverse image and video processing challenges.



```
 5 Using multiprocessing: 8 cpus detected, using 4
   processes
 6 Process 0: (0, 70)
 7 Process 1: (64, 134)
 8 Process 2: (128, 198)
 9 Process 3: (192, 262)
10 GPU Soft MUSICAL
11 GPU Soft MUSICAL
12 GPU Soft MUSICAL
13 GPU Soft MUSICAL
```

Figure 13: OUTPUT



```
30 Time processing pixels: 42.31667423248291 total and 0
   .002583 s per pixel
31 Time processing pixels: 42.502440452575684 total and
   0.002594 s per pixel
32 Time processing pixels: 42.558337926864624 total and
   0.002598 s per pixel
33 Time processing pixels: 42.81330847740173 total and 0
   .002613 s per pixel
34 Image done.
35 Elapsed time (seconds): 55.09667634963989
36 Elapsed time: 0:00:55.09667634963989
```

Figure 14: OUTPUT

The output shows the code using multiple processors (CPUs) to analyze an image in parallel.
Lines 5-9: The code detects 8 CPUs and uses 4 processes to analyze the image simultaneously.

Lines 6-8: Each process is assigned a section of the image to analyze (rows in this case).
Lines 30-33: These lines report how long it took each process to analyze its assigned section.
Lines 34-36: The image analysis is complete, and it took about 55 seconds.

6.8 FPGA VS GPU in Image/Video Processing

FPGA (Field-Programmable Gate Array) and GPU (Graphics Processing Unit) are two distinct hardware platforms with unique architectures, each offering advantages and limitations for image and video processing applications.
FPGAs are highly configurable integrated circuits that can be reprogrammed to perform specific tasks. Their parallel architecture enables concurrent execution of multiple processing tasks, making them suitable for real-time image and video processing tasks. FPGAs offer low latency and high throughput, making them ideal for applications requiring fast response times, such as surveillance systems, medical imaging, and industrial inspection. Additionally, FPGAs consume less power compared to GPUs, making them suitable for embedded and low-power applications.

On the other hand, GPUs are specialized processors designed primarily for graphics rendering but have evolved to handle general-purpose parallel computing tasks, including image and video processing. GPUs offer high computational throughput and are well-suited for parallelizable tasks such as convolution, filtering, and matrix operations. They excel at tasks requiring complex mathematical computations and are widely used in applications such as computer vision, machine learning, and multimedia processing. However, GPUs may suffer from higher latency compared to FPGAs, especially in applications with strict real-time requirements.

In terms of programming complexity, FPGA development typically requires hardware description languages (HDL) such as Verilog or VHDL, along with specialized design tools like Vivado HLS. FPGA development cycles may be longer due to the need for hardware optimization and testing.[3] In contrast, GPUs are programmed using high-level languages like CUDA or OpenCL, which are easier to learn and use. GPU development cycles are typically shorter, allowing for faster prototyping and iteration.
FPGAs offer low latency, high throughput, and low power consumption, making them suitable for real-time and embedded image/video processing applications. GPUs, on the other hand, excel at parallelizable tasks, offering high computational throughput and ease of programming, making them ideal for general-purpose image/video processing and machine learning applications. The choice between FPGA and GPU depends on the specific requirements of the application, including performance, power consumption, latency, and development complexity.[3]

7.DIFFICULTIES IN THE PROCESS

1.Limited Learning Resources:The absence of comprehensive learning materials and resources pertaining to FPGA technology posed a significant challenge. Finding adequate documentation, textbooks, or online tutorials that offer a thorough understanding of FPGA concepts proved to be challenging. This limited availability hindered the depth of theoretical knowledge and practical application.
2.Lack of Structured Guidance: A notable challenge during the learning journey was the lack of structured guidance or mentorship. The absence of a clear roadmap or mentor to provide step-by-step direction in understanding and applying FPGA concepts created uncertainties. A structured guidance system could have

streamlined the learning process and improved overall comprehension.
3.Limited Access to Support:Another obstacle was the limited access to timely support or assistance when facing challenges. The absence of immediate help or a reliable support system hindered the ability to overcome specific roadblocks efficiently. Timely access to support could have accelerated problem-solving and improved the overall learning experience.
4.Conceptual Complexity:Navigating the intricate conceptual landscape of MUSICAL Algorithm presented a notable challenge. Understanding the working,calculations and flow of the algorithm involved grappling with abstract concepts that required a deep level of comprehension.

8.APPLICATIONS

1)Biomedical Imaging: The MUSICAL Algorithm implementation can aid in signal classification for high-resolution microscopy images, facilitating cellular analysis and disease diagnosis in biomedical imaging.
2)Computer Vision: Sobel edge detection can be utilized in various computer vision applications such as object recognition, image segmentation, and feature extraction.
3)Multimedia Processing: Pattern generation techniques can enhance multimedia applications by generating dynamic visual effects in videos, animations, and presentations.
4)Video Editing: Video cropping algorithms can be employed in video editing software to remove unwanted regions or resize video frames, enhancing the editing process and improving the final output.
5)Content Analysis: The ability to crop videos and images can facilitate content analysis tasks such as object tracking, motion detection, and scene recognition.

6)Real-time Image Processing: FPGA-based implementations offer high-speed and parallel processing capabilities, making them suitable for real-time image processing applications in industries like surveillance, automotive, and robotics.
7)Medical Imaging: Edge detection and pattern generation techniques can be applied in medical imaging for tasks such as tumor detection, organ segmentation, and anatomical analysis.
8)Industrial Inspection: Image processing techniques can aid in quality control and defect detection in manufacturing processes, ensuring product quality and efficiency.
9)Remote Sensing: Image processing algorithms can be utilized in remote sensing applications for analyzing satellite images, environmental monitoring, and land use classification.

9.VALIDATION

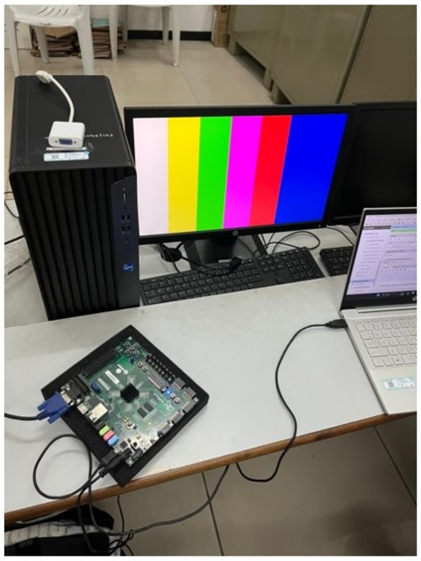


Figure 15: IMPLEMENTATION ON FPGA

10.FUTURE SCOPE

We have done a simulation project.So by exploring Domain-Specific Applications, our model can be extended to be deployed to those applications on real time.For example, we could investigate applying your model to medical imaging tasks, industrial inspection, or agricultural applications.Applying our model to real time scenarios can be implemented by using FPGA connected to camera.
We could investigate options for integrating voice commands, gesture recognition, or tactile feedback to facilitate interaction with the system and consider designing a user-friendly interface or exploring accessibility standards, such as incorporating support for screen readers or Braille displays.

Also,we could investigate the possibility of deploying our object classification model on edge devices, such as smartphones, IoT devices, or embedded systems and Optimize the model size and computational requirements to ensure efficient execution on resource-limited devices.

11.CONCLUSION

In conclusion, the exploration of FPGA technology,video/image processing, and application testing on Xilinx Vivado has been a multifaceted journey. The challenges encountered, ranging from conceptual complexities to limitations in learning resources, have underscored the intricacies of delving into advanced technologies. Despite these challenges, the endeavor has resulted in a profound understanding of FPGA fundamentals, their applications in processing high-resolution multimedia, and the practical utilization of Xilinx Vivado for testing and validation. The significance of FPGA technology in diverse applications, from digital signal processing and communication systems to medical imaging and robotics, highlights its versatility and adaptability in addressing complex real-world problems. The reconfigurable nature of FPGAs empowers developers to tailor solutions to specific requirements, offering a level of flexibility not easily achievable with traditional hardware.
The journey has emphasized the critical role of hands-on experience, practical implementation, and the need for accessible learning resources. Navigating the complexities of FPGA architecture has provided insights into the dynamic world of programmable logic and its impact on various industries.
As technology continues to evolve, the skills acquired in this exploration position us at the intersection of innovation and application. The knowledge gained is

not only applicable in current scenarios but also serves as a foundation for adapting to emerging technologies in the rapidly evolving landscape of digital design and multimedia processing.
In essence, this exploration has been a valuable chapter in the ongoing quest for technological proficiency. It reinforces the idea that overcoming challenges is an integral part of the learning process, and each hurdle surmounted contributes to a more comprehensive understanding of advanced technologies. As we move forward, the insights gained from this endeavor will undoubtedly serve as a solid foundation for continued exploration and innovation in the dynamic realm of FPGA technology.

12.REFERENCES

1)https://www.researchgate.net/publication/311066897
2)https://support.xilinx.com/s/article/862633?language
3)https://semiengineering.com/developing-4k-video-projects-with-fpgas
4)https://www.sciencedirect.com/science/article/abs/pii/

13.TEAM MEMBERS

K.P Sai Swetha	**1602-21-735-109**
G.Sri Ganga Pranav	**1602-21-735-117**