中国科学技术大学

SSE PRACTICE SYNTHESIS

Design and Implementation of a Hardware-Accelerated Image Filtering System using Zynq FPGA

Course Code: EIEN6711P
Group: FPGA Image Processing-2
Teacher Name: Dr. Ding Qing
Date: 2025 June 30

WELCOME

中国科学技术大学

Team Members

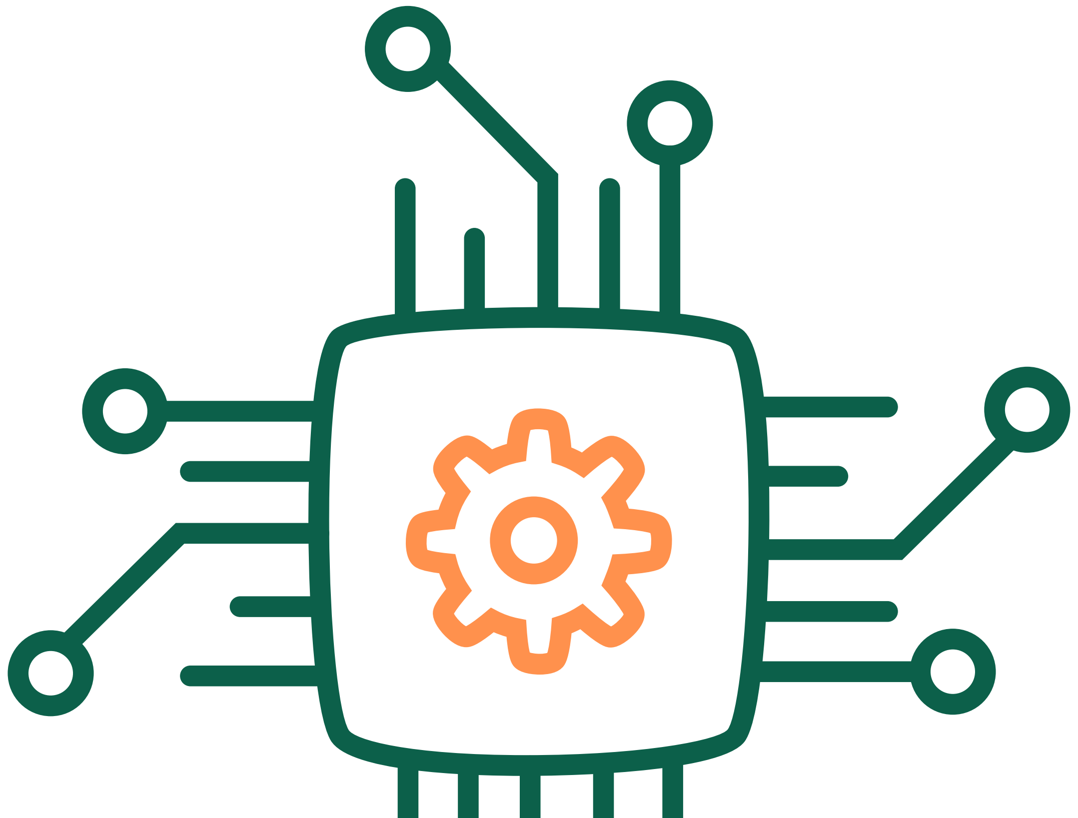
RAOHA BIN MEJBA (李一含)
Student ID: SL24225002

Student Name: MARJAN UR RAHMAN REMO (雷漠)
Student ID: SL24225005

MD HEZBULLAH (李龙)
Student ID: SL24225009

中国科学技术大学

Table Of Content



中国科学技术大学

Introduction

Project Overview:

- In this project, we developed a real-time image processing system using the ZedBoard Zynq-7000 FPGA. This board integrates a dual-core ARM Cortex-A9 processor (PS) with programmable logic (PL) on a single chip. This SoC architecture enables a flexible hardware-software co-design approach—software controls the system while hardware performs high-speed image filtering, achieving efficient and scalable performance.

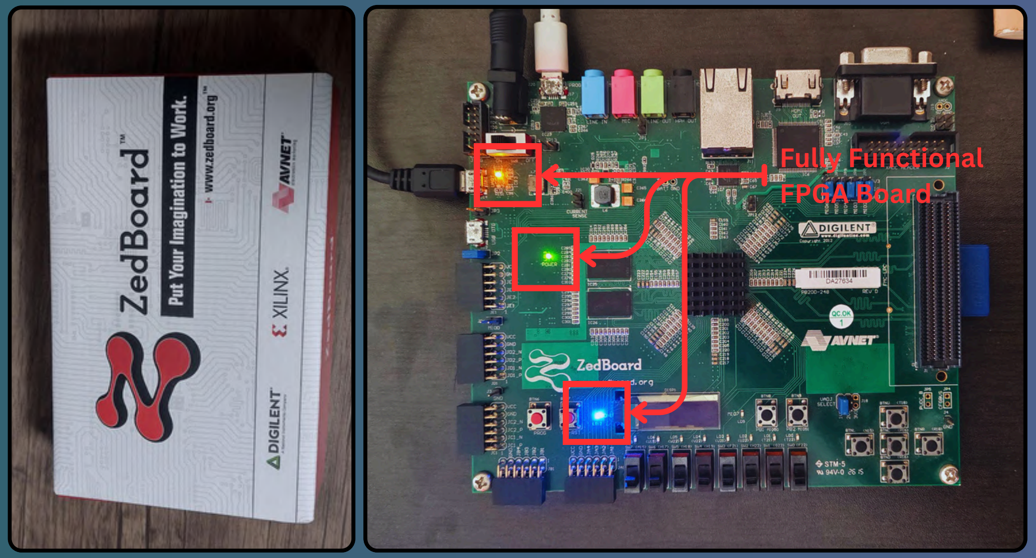
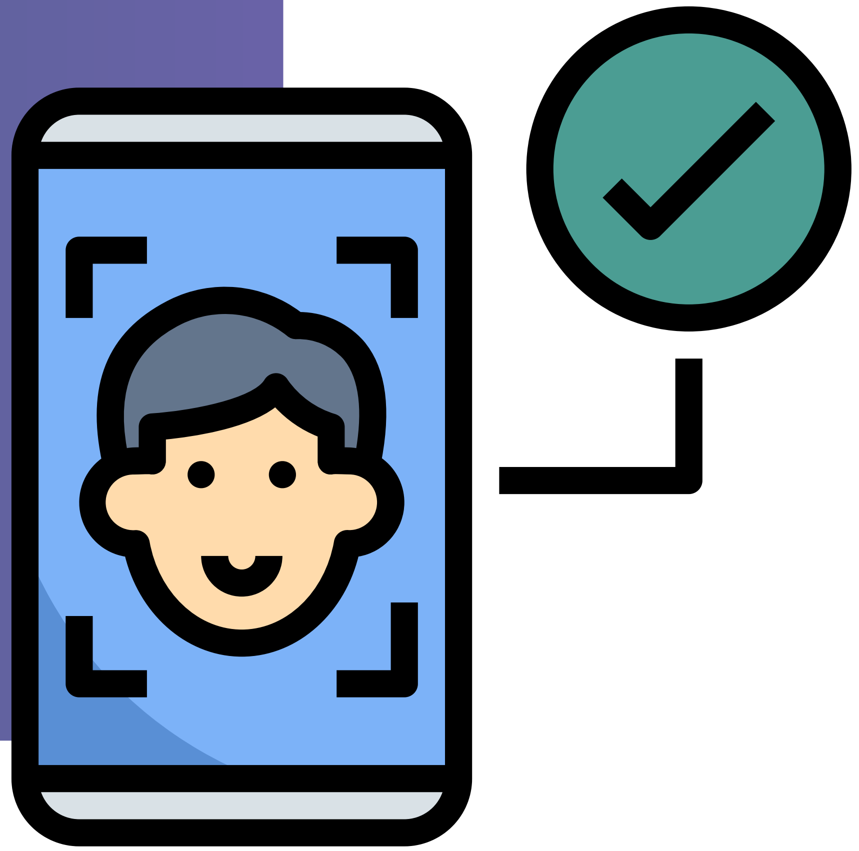


Fig 1: ZedBoard Zynq-7000 Development Kit used in the project.



Introduction

Project Goal:

- The primary goal of our project is to apply five different image filtering operations on grayscale images using FPGA hardware acceleration. The filters implemented in our system are:

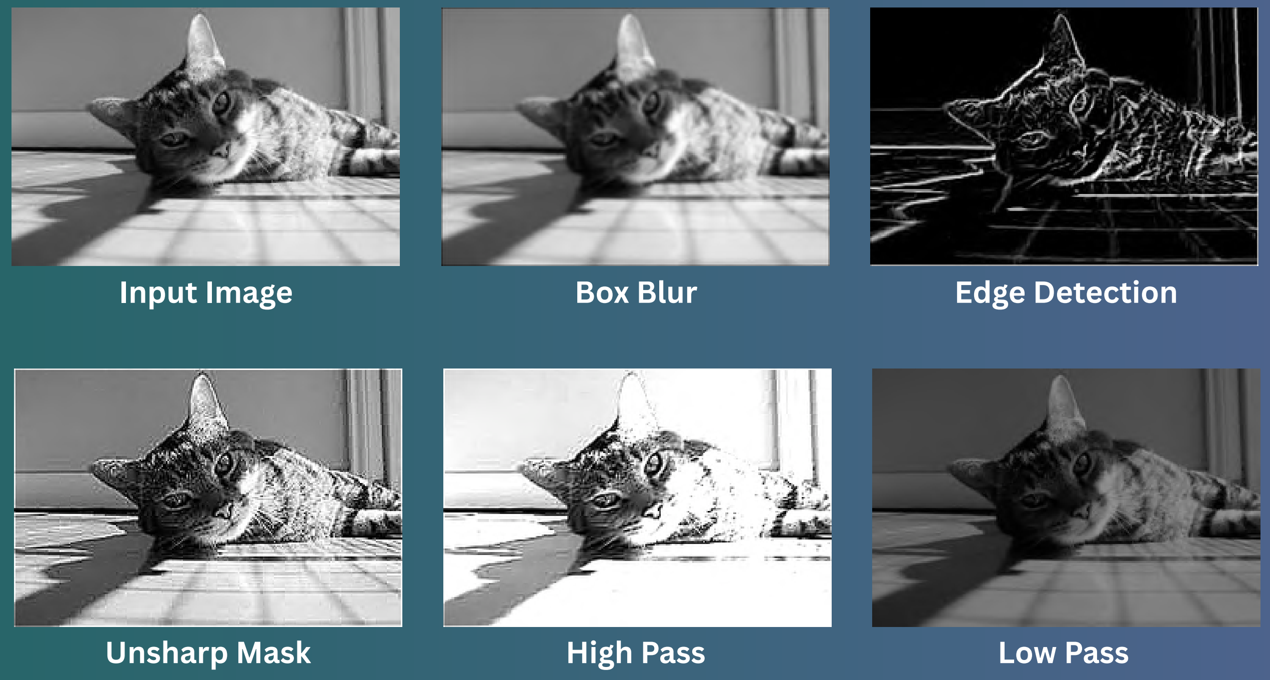
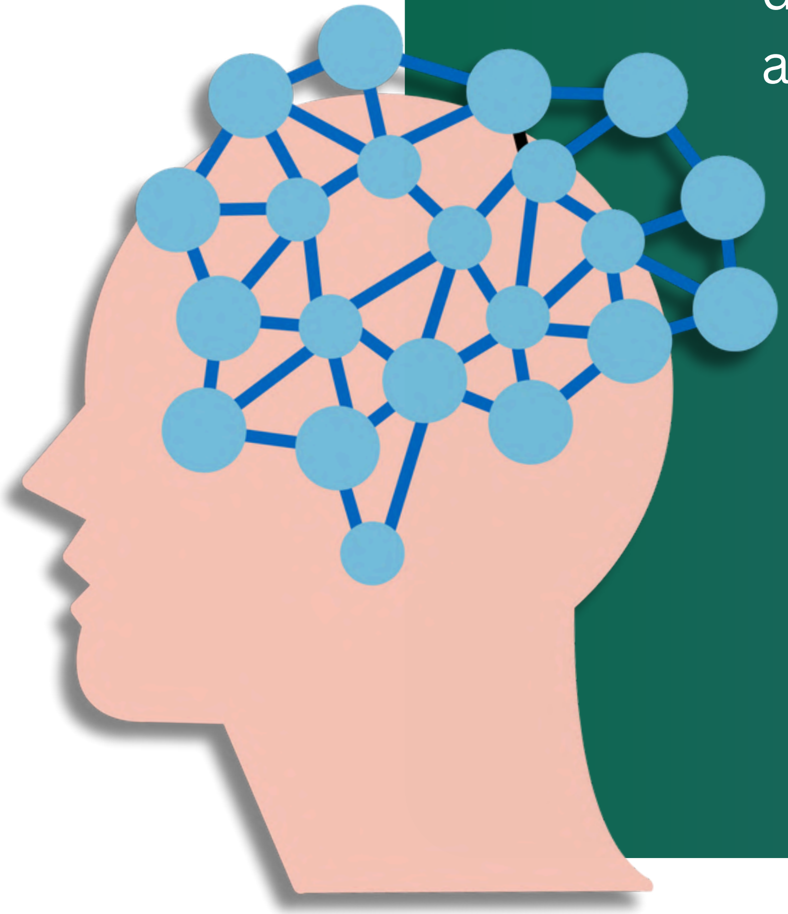


Fig 2: Example of input image and five filtered outputs Image we get from the GUI App.

Image Filter Logic

Kernel Convolution

- Kernel convolution is a fundamental technique in image processing used to modify pixel values based on their neighbors. It works by sliding a small matrix, called a kernel or filter, over the image. At each position, the kernel values are multiplied with the corresponding image pixels, and the results are summed to produce a new pixel value. This operation helps detect edges, smooth images, sharpen details, or enhance specific features depending on the kernel used. It forms the core of many filtering algorithms and is essential for tasks like blurring, edge detection, and noise reduction.



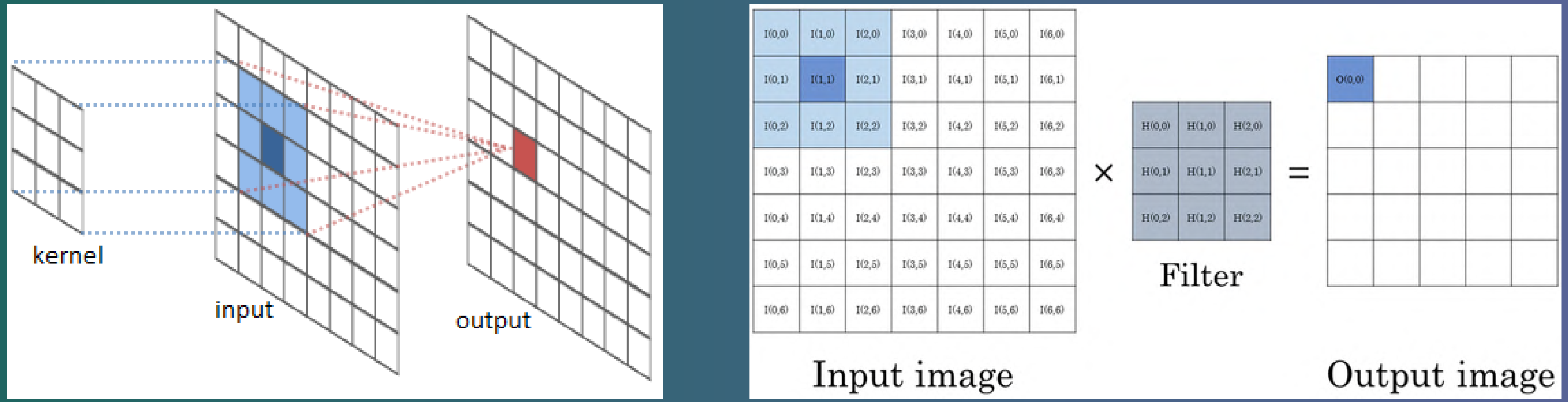


Fig 3: Image Processing technique using kernel convolution.

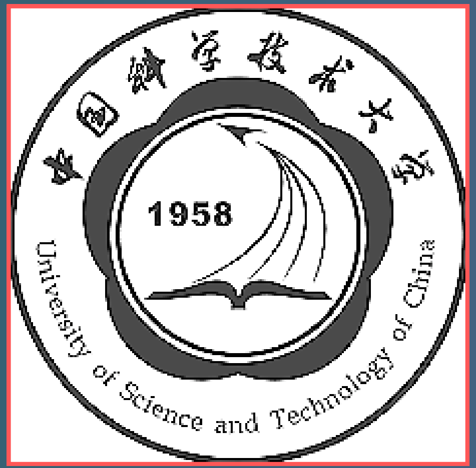
中国科学技术大学

Image Filter Logic

Box Blur Filter

- The Box Blur filter is used to smooth an image by averaging the values of neighboring pixels. It reduces noise and small details, making the image appear soft and less sharp.

$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$



Original Image

Box Blur

Fig 4: Example of Box Blur filter applied to the input image.

中国科学技术大学

Image Filter Logic

Edge Detection (Sobel Operator)

- Edge detection is used to highlight the boundaries between objects in an image. This project uses the Sobel operator, which calculates gradients in both horizontal and vertical directions.

$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

G(x)

$$\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

G(y)

$$G = \sqrt{G_x^2 + G_y^2}$$

Original Image

Edge Detection

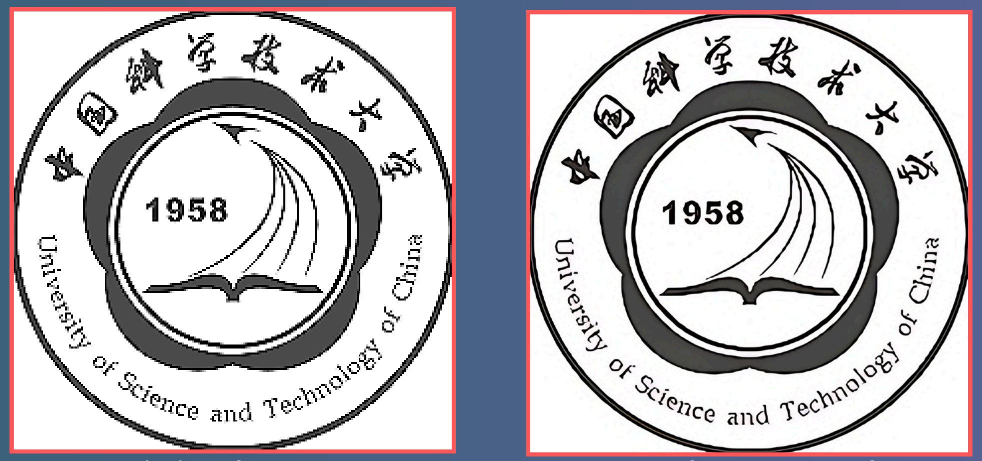
Fig 5: Image after applying Edge Detection using Sobel filter.

Image Filter Logic

Unsharp Masking

- Unsharp masking enhances the sharpness of an image by subtracting a blurred version from the original image. It increases contrast around edges, making them more visible.

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$



Original Image   Unsharp Mask

Fig 6: Result of Unsharp Masking filter.

Image Filter Logic

High Pass Filter

- A high pass filter emphasizes high-frequency components like edges and fine textures. It suppresses low-frequency content like smooth regions and background areas.

$$\begin{bmatrix} 1 & -\frac{7}{9} & -\frac{7}{9} \\ 0 & 5 & 0 \\ 0 & -\frac{7}{9} & -\frac{7}{9} \end{bmatrix}$$

Original Image

High Pass

Fig 7: Image after High Pass Filtering.

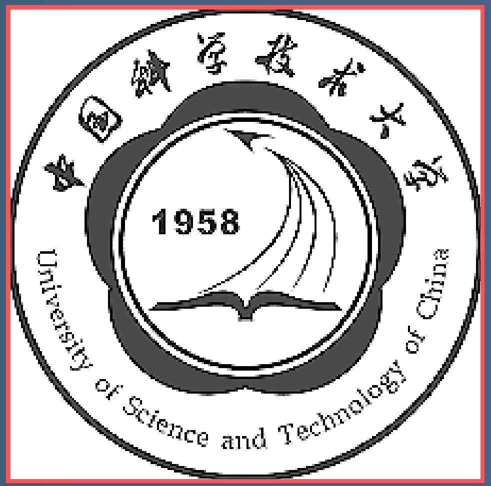
中国科学技术大学

Image Filter Logic

Low Pass Filter

- Unsharp masking enhances the sharpness of an image by subtracting a blurred version from the original image. It increases contrast around edges, making them more visible.

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & \frac{5}{9} & 0 \\ 0 & 0 & 0 \end{bmatrix}$$



Original Image

Low Pass

Fig 8: Output of Low Pass Filtering on the input image.

Methodology

Tools Used

- This combination of tools and technologies allowed us to design a reliable, fast, and user-friendly image processing platform. Each component was carefully selected to ensure seamless integration between the hardware and software layers of the system.

Category	Tool / Language	Purpose
FPGA Design	Vivado 2024.2	Block design, bitstream generation
IP Development	Vitis HLS 2024.2	Custom filter IP design
Embedded Software	Vitis IDE (C Language)	UART, DMA, IP control
GUI Development	Python (Tkinter, Pillow, PySerial)	User interface, image display, UART communication
Communication Interface	UART (115200 bps, COM4)	Host ↔ FPGA data transfer
Image Format	128×128, 8-bit grayscale BMP	Fast processing and transfer

Internal Hardware Design (Block Diagram)

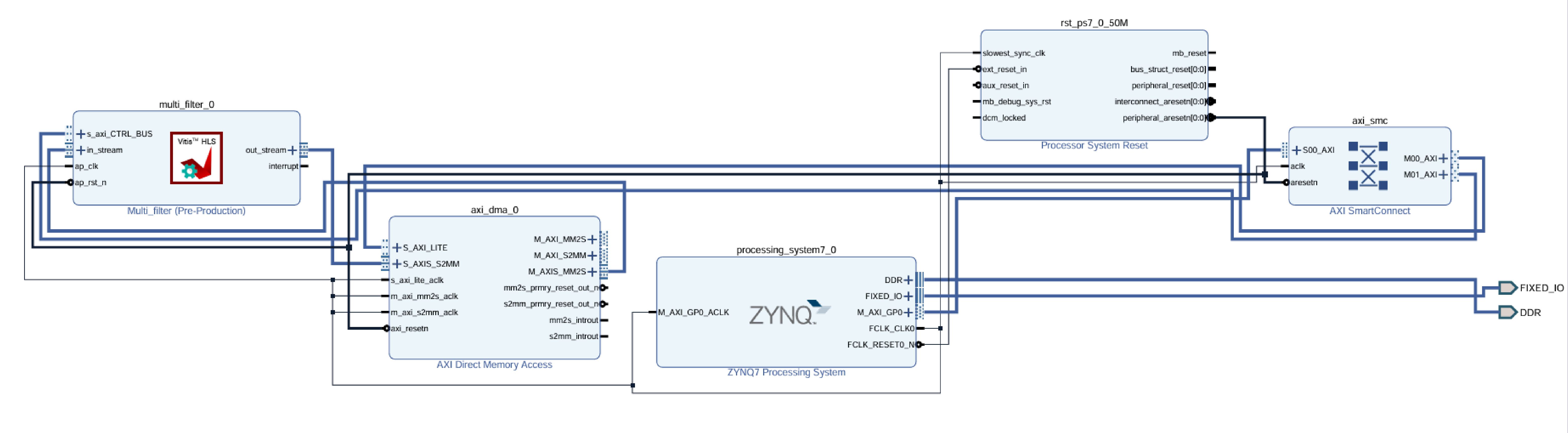


Fig 9: Block diagram of the hardware design in Vivado. This figure shows the Zynq PS, AXI DMA, SmartConnect, and multi_filter IP, along with their connections.

Methodology

Control Logic Flow

The overall flow of operations on the FPGA side is managed using simple control logic:

1. UART receives image bytes and stores them in DDR.
2. DMA MM2S is configured to send the image to the IP.
3. IP processes the image and streams five filtered results.
4. DMA S2MM stores each result in memory.
5. After all results are available, they are sent back over UART.

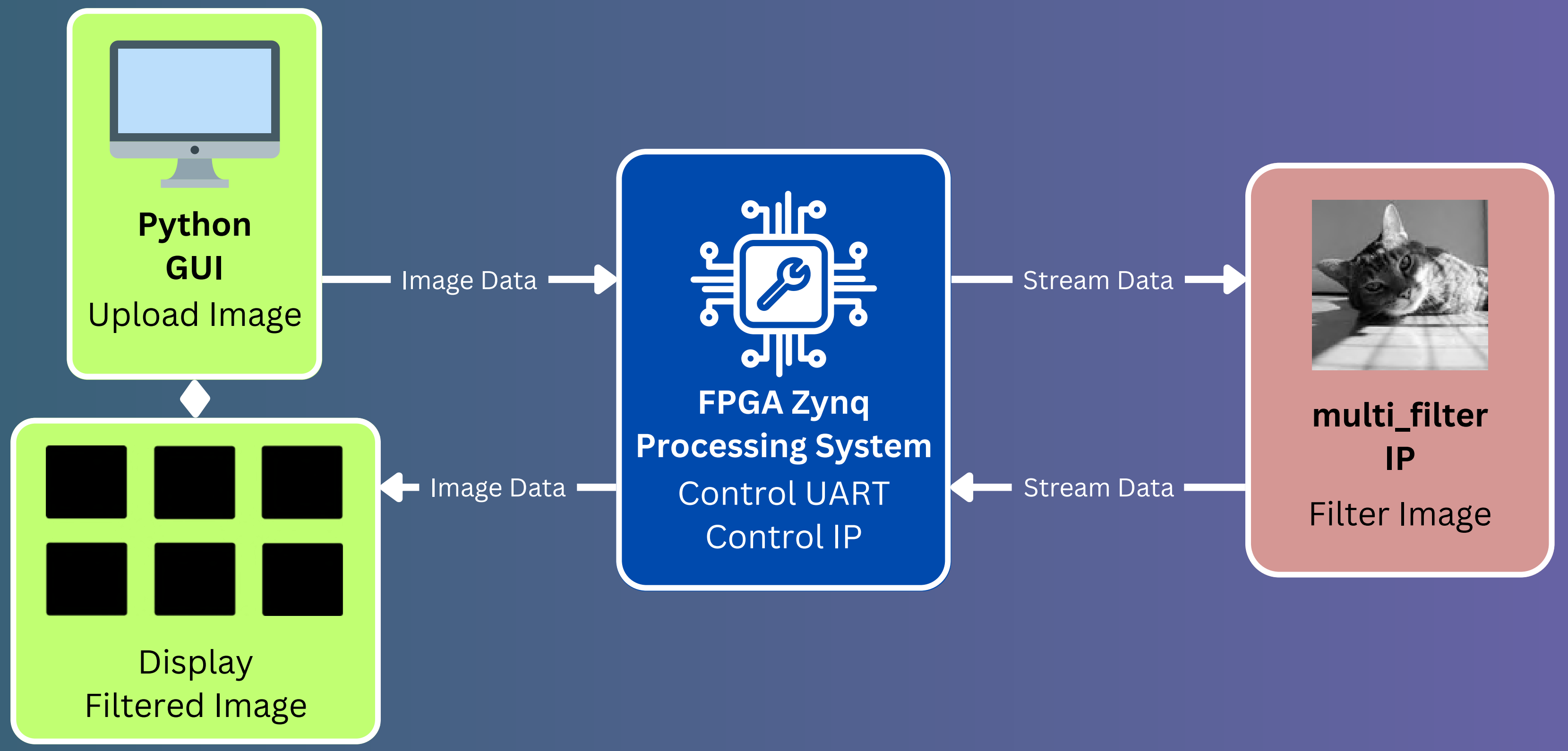


Fig 10: General data flow between Python GUI, Zynq PS, DMA, and multi_filter IP.

User Interface f GUI APP

Fig 10: User Interface of Python GUI App.

Result Analysis

Simulation Waveform Analysis

- We validated the multi_filter IP core using behavioral simulation in Vitis HLS. A C testbench was written to provide synthetic input and verify the output.

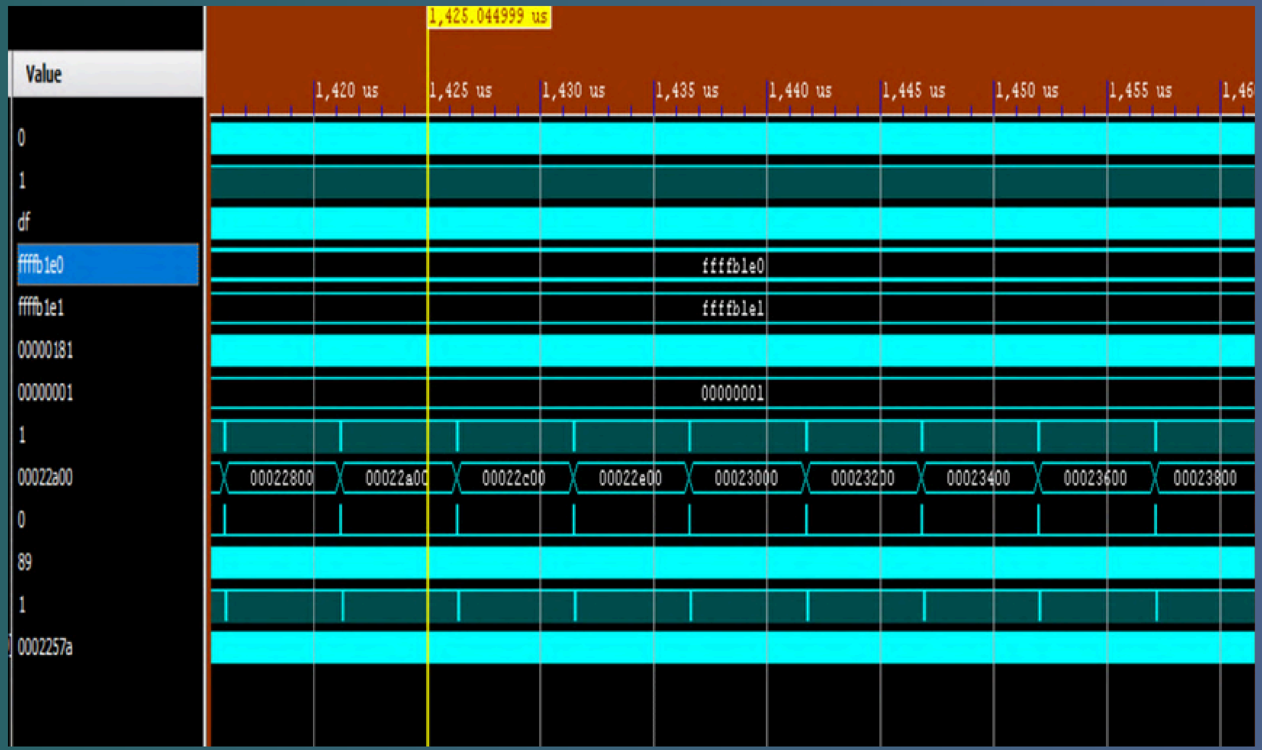


Fig 11: Simulation waveform showing valid data signals, image flow, and interrupt behaviour.

中国科学技术大学

Experiment & Analysis

Hardware Resource Utilization

- We also analyzed the FPGA resource usage after synthesis and implementation.

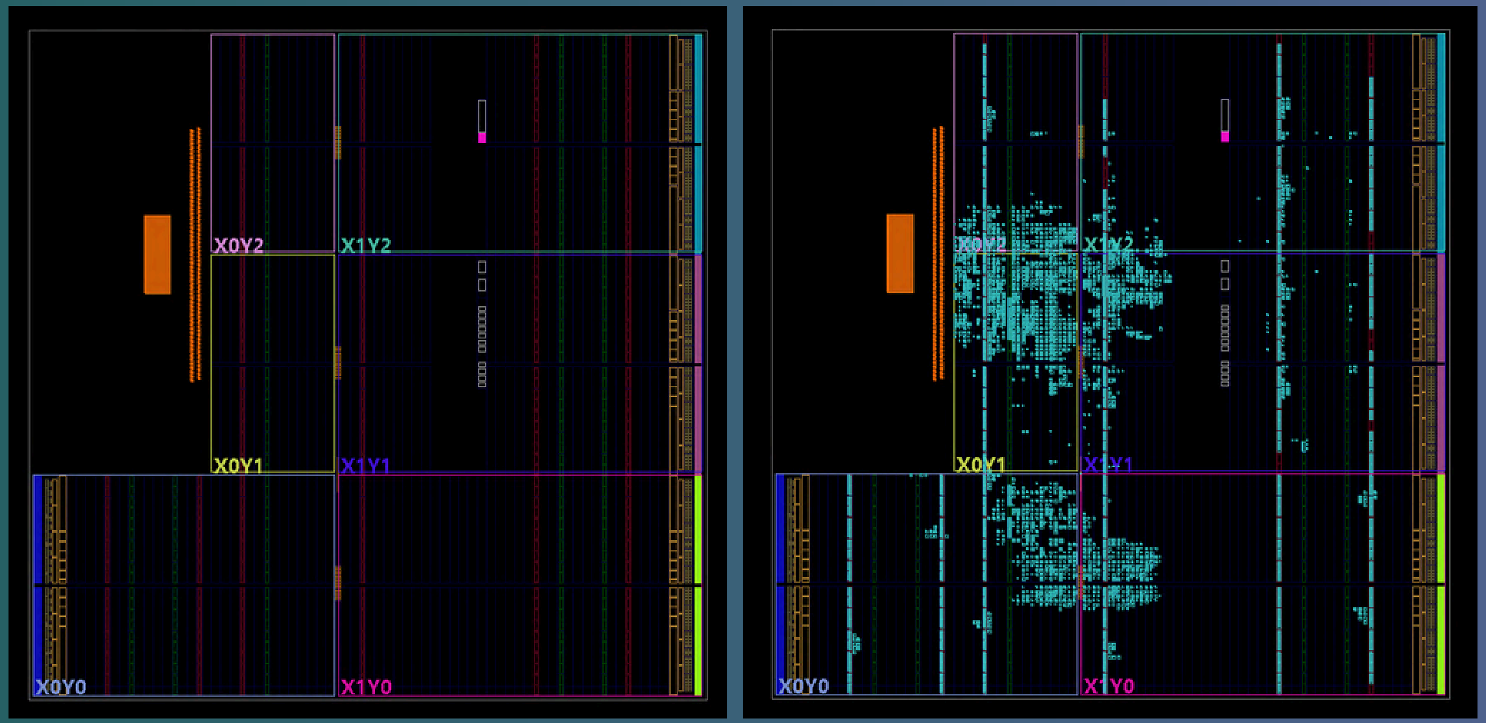


Fig 16: Synthesis and implementation layout in Vivado.

Result Analysis

Physical Hardware Validation

- We validated the final implementation by flashing the bitstream to the ZedBoard using JTAG and running the software application from Vitis. The board showed proper activity with LED indicators and terminal logs.put and verify the output.

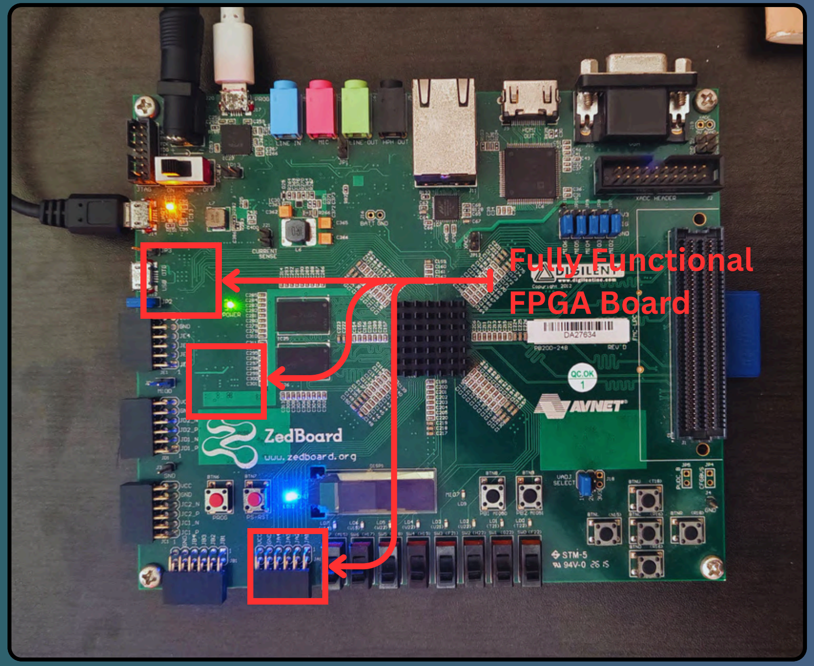
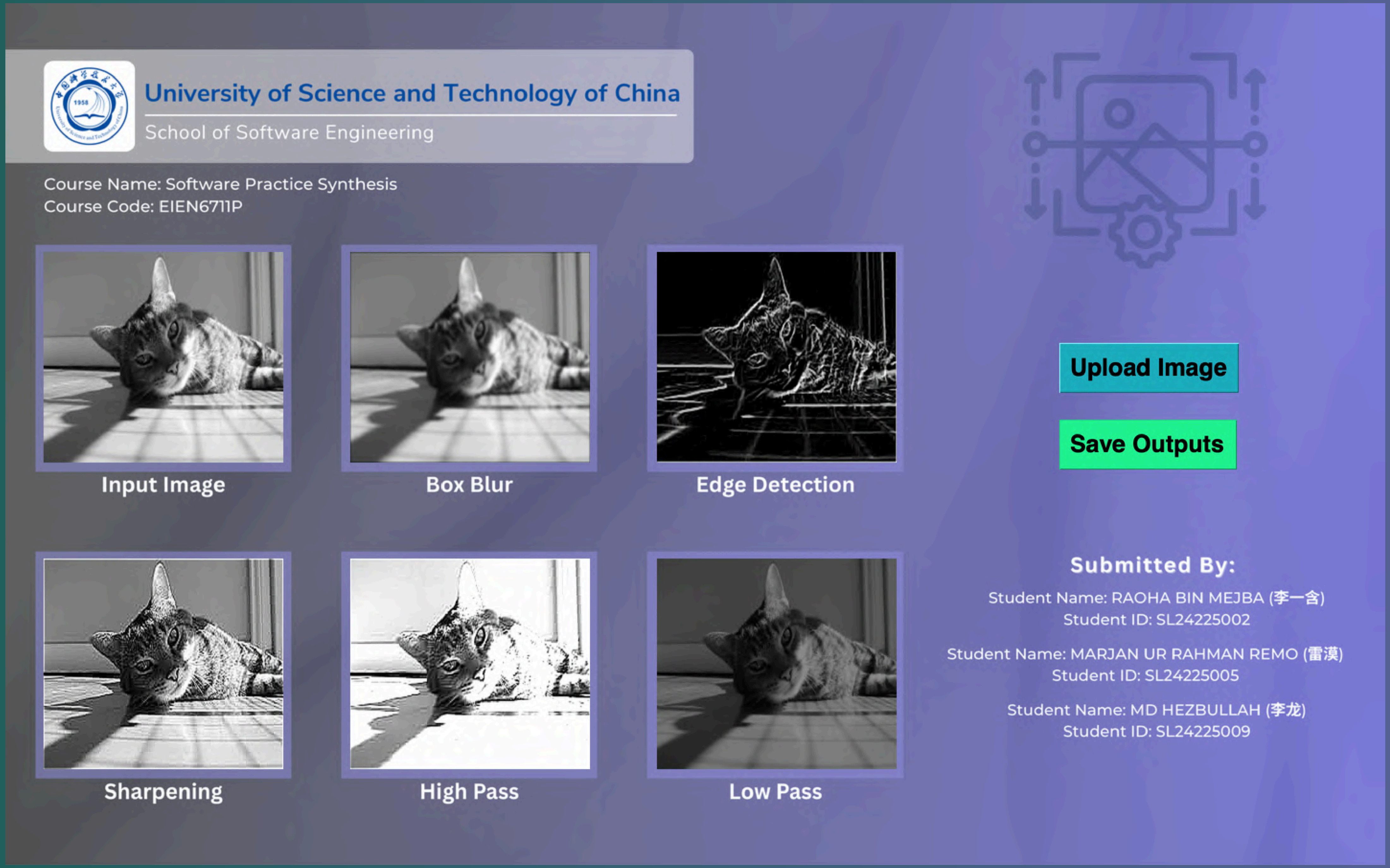


Fig 17: ZedBoard running the image filtering system (powered on, connected to PC). This confirmed that the board was correctly executing the image processing pipeline and communicating with the PC.

中国科学技术大学

Project Screenshots

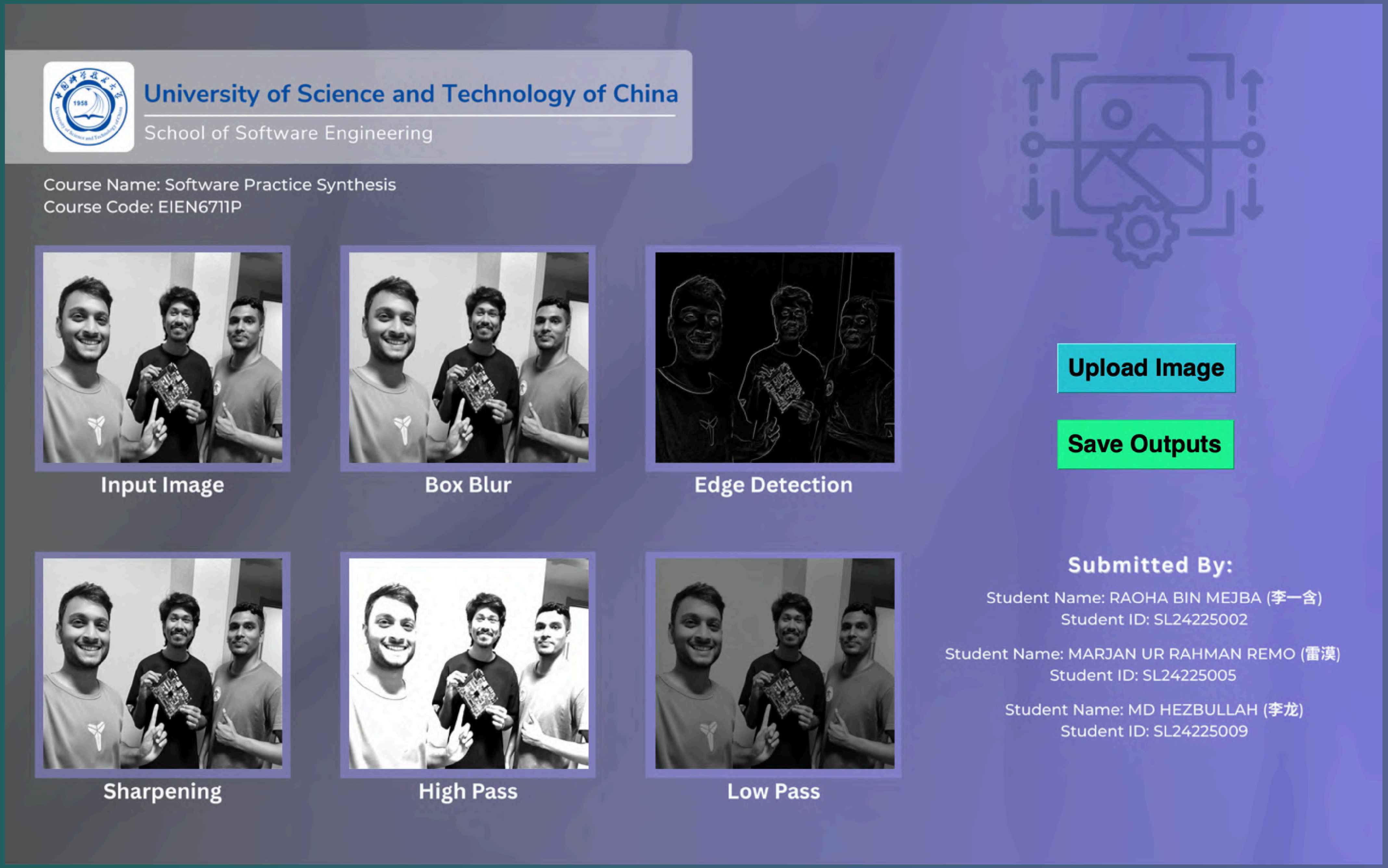


中国科学技术大学

Project Screenshots

中国科学技术大学

Project Screenshots



中国科学技术大学

Project Screenshots

中国科学技术大学

Result Analysis

Output Verification Summary

- All filters passed visual and data-level verification for correctness and consistency.

Test Image	Output	Expected Behavior	Observed Behavior
Input_Image_1	Edge Map	Sharp outlines	Matched expectation
Input_Image_5	Box Blur	Soft, low contrast	Blurred successfully
Input_Image_4	Unsharp	Enhanced details	Clear sharpening seen
Input_Image_2	Low Pass	Smooth background	Reduced sharpness
Input_Image_3	High Pass	Highlighted edges	Bright outlines visible

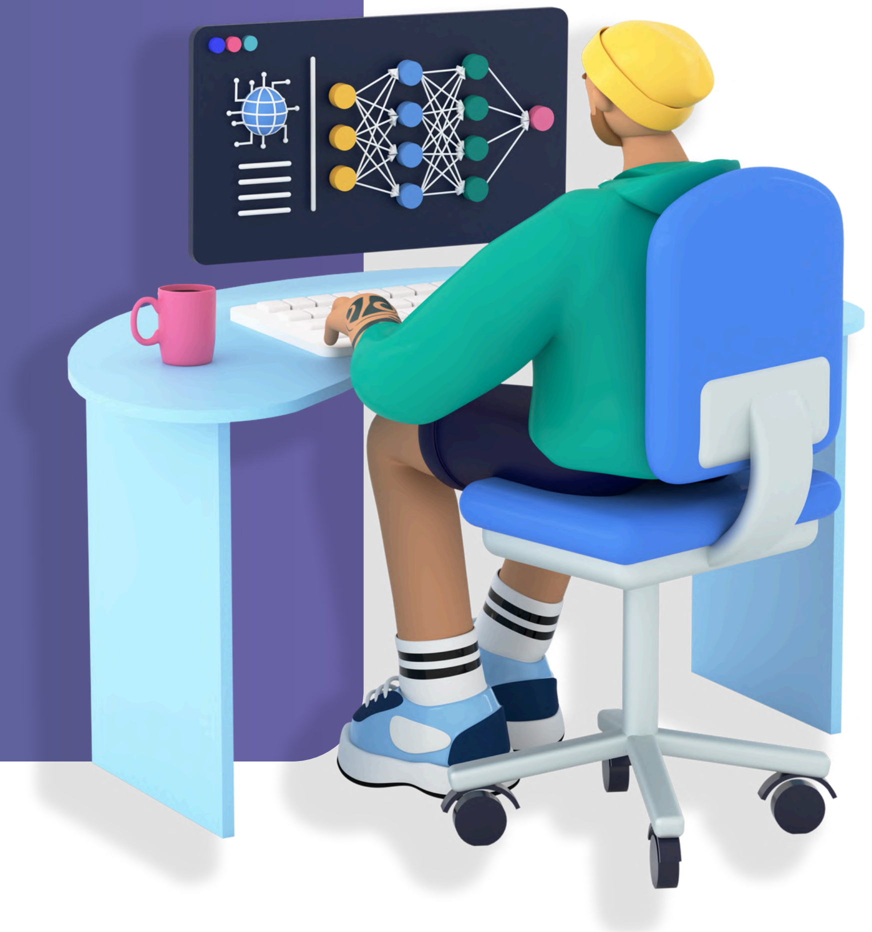
Conclusions & Discussion

Key achievements

- Real-time image filtering using FPGA
- GUI-based interactive platform
- All filters integrated in one IP

Future Scope

- Add video filtering
- Support color (RGB) images
- Real-time streaming/webcam input



Thank You