Project Report: FPGA-Based Spatial Filtering for Image Processing

By- Atharva Wasade

B.E. Electronics & Communication Engineering,

Birla Institute of Technology & Science, Pilani Hyderabad Campus

ID-2022AAPS0469H

Github

1. Objective

The objective of this project is to design, simulate, and implement a spatial filtering IP core on an FPGA. The design processes real-time image data using convolution-based filters (such as edge detection, blur, or sharpening) which enables hardware-accelerated image enhancement on the Xilinx Zynq platform.

2. Tools & Technologies

- HDL Language: Verilog
- FPGA Platform: Xilinx Zynq (e.g., Zynq-7000 series)
- Development Tool: Xilinx Vivado
- Simulation: Vivado Testbench (tb.v)
- Constraint File: outputBuffer.xdc

3. System Overview

The system implements a hardware image filter using:
- A line buffer for holding rows of image data
- A convolution engine to apply a kernel (e.g., 3x3)
- Control logic to manage synchronization and valid signals
- A top-level module (imageProcessTop.v) that integrates all components

This design allows the system to operate in real-time for streaming video or camera input.

4. Modules and Descriptions

- lineBuffer.v:

Stores multiple lines of the incoming image. Implements a sliding window required for convolution. Critical for accessing neighboring pixels in a streaming architecture.

- conv.v / conv1.v:

Perform the convolution (multiply-and-accumulate). Apply user-defined kernel weights to the 3x3 image window. Used for filters like edge detection, smoothing, etc.

- imageControl.v:

Generates control signals for synchronization. Manages pixel validity and data readiness. Coordinates between the line buffer and convolution engine.

- imageProcessTop.v:

Top-level wrapper module. Instantiates the line buffer, convolution unit, and control unit. Provides the final processed image output.

- outputBuffer.xdc:

FPGA constraints: maps internal signals to actual FPGA I/O pins. Also includes timing constraints (if needed).

- tb.v:

Testbench to simulate the entire system. Provides sample image pixel data. Verifies correctness of convolution output.

5. Workflow

1. Design
   - Write and verify individual HDL modules.
   - Use parameterized and reusable logic.
2. Simulation
   - Use tb.v to simulate various inputs and validate functionality.
3. Synthesis and Implementation
   - Compile and synthesize using Vivado.
   - Perform place-and-route.
4. Bitstream Generation
   - Generate .bit file for FPGA configuration.
5. Deployment
   - Interface with Zynq PS (Processing System) via AXI (if applicable).
   - Display or store filtered image output.

6. Applications

- Edge detection in video feeds
- Preprocessing for computer vision and AI

- Real-time image enhancement
- Embedded camera systems in surveillance or robotics

7. References

- Line Buffer Design: https://youtu.be/n35zS__YEFQ
- MAC Unit Design: https://youtu.be/6El_NQrpgCY
- Control Logic: https://youtu.be/v8pHH-q-0sE
- Edge Detection: https://youtu.be/TcjqZG2pbHw
- System Integration: https://youtu.be/r45dkUHIbk4
- Simulation Walkthrough: https://youtu.be/tO5pZ2K9U9I

8. Conclusion

This project successfully demonstrates how spatial filtering can be implemented using Verilog on an FPGA. The modular and scalable design allows the filter to be reused or extended to different kernel sizes or image resolutions. Hardware acceleration via FPGA allows real-time performance, making it highly suitable for embedded vision applications.