Summary of Research Paper

Underwater Image Enhancement Using FPGA-Based Gaussian Filters with Approximation Techniques
Summarized on August 2, 2025

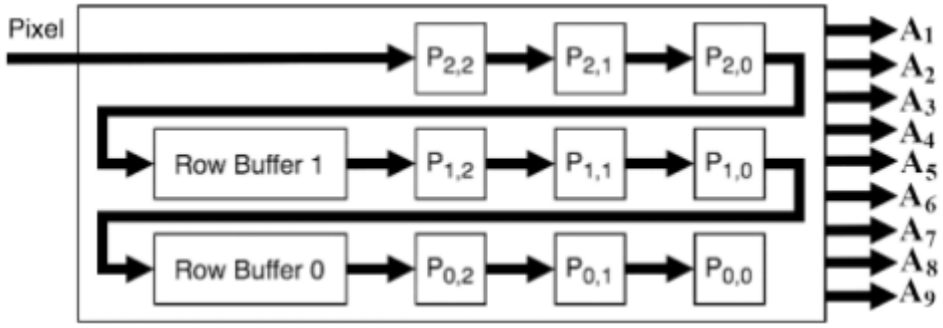
1. **Introduction / Abstract**
   The paper addresses the problem of poor visual quality in underwater images caused by natural effects like light absorption and scattering, which create haziness and noise. To improve these images, the authors propose an efficient method using a **Gaussian filter** (a mathematical tool for smoothing images and reducing noise) implemented on an **FPGA** (a type of reconfigurable hardware device that allows fast, parallel processing). The key innovation is a pipeline architecture combined with approximate adders (simplified arithmetic units that trade some accuracy for better speed and lower power use). The results show over 150% speed improvement and more than 34% power reduction, with some increase in hardware area. This trade-off suits error-tolerant applications like image and video processing.

2. **Methodology**
   The researchers designed a pipeline Gaussian filter architecture on an FPGA, which processes image pixels in stages to increase throughput. They used line buffers and window buffers to efficiently handle pixel data for convolution (a process where the filter mask moves over the image to smooth it). To reduce power and increase speed, they incorporated various approximate adders that simplify addition operations by allowing minor errors. They tested ten different approximate adder designs, applying approximations mainly to the least significant bits of the adders. The system was simulated in MATLAB with real underwater images corrupted by Gaussian noise, and then synthesized on an Intel MAX10 FPGA to measure power, speed, and area.

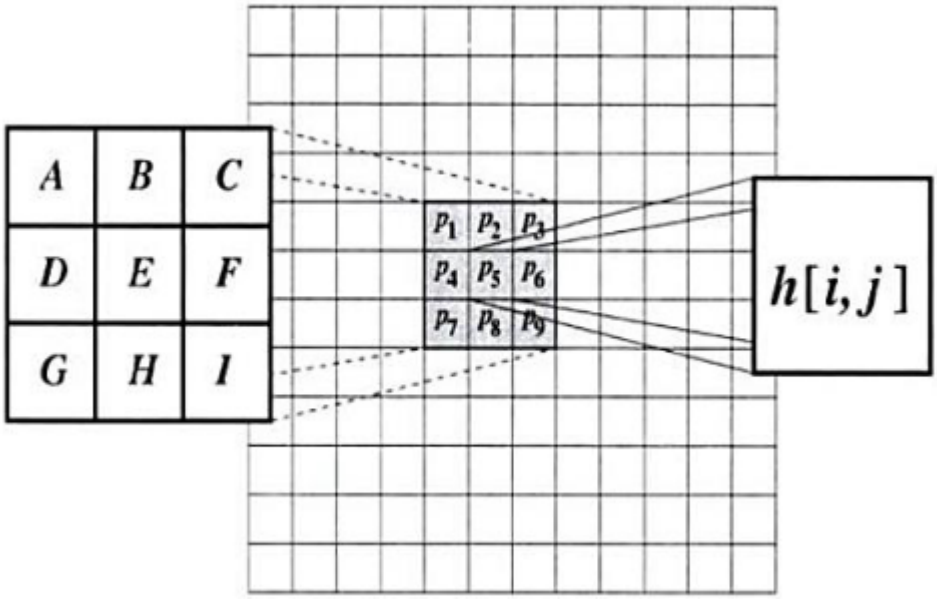
3. **Theory / Mathematics**
   The Gaussian filter is based on the two-dimensional Gaussian function:
   [f(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2 + y^2}{2\sigma^2}}]
   where (\sigma) controls the spread of the smoothing effect. This function weights pixels near the center more heavily, smoothing the image and reducing noise. The filter is separable, meaning a 2D convolution can be done as two 1D convolutions, improving efficiency. The convolution operation for a 3×3 kernel is expressed as:
   [h[i,j] = \sum_{k=1}^9 W_k P_k]
   where (W_k) are kernel weights and (P_k) are pixel values in the window.
   Image quality metrics used include:

- **PSNR (Peak Signal-to-Noise Ratio):** Measures how close the filtered image is to the original, higher is better.
- **MSE (Mean Square Error):** Average squared difference between images, lower is better.
- **SSIM (Structural Similarity Index):** Assesses perceived image quality considering luminance, contrast, and structure.
- **Error Distance (ED), Mean Error Distance (MED), Normalized Error Distance (NED):** Quantify spatial differences between images.
  These metrics help evaluate the trade-off between approximation-induced errors and image quality.

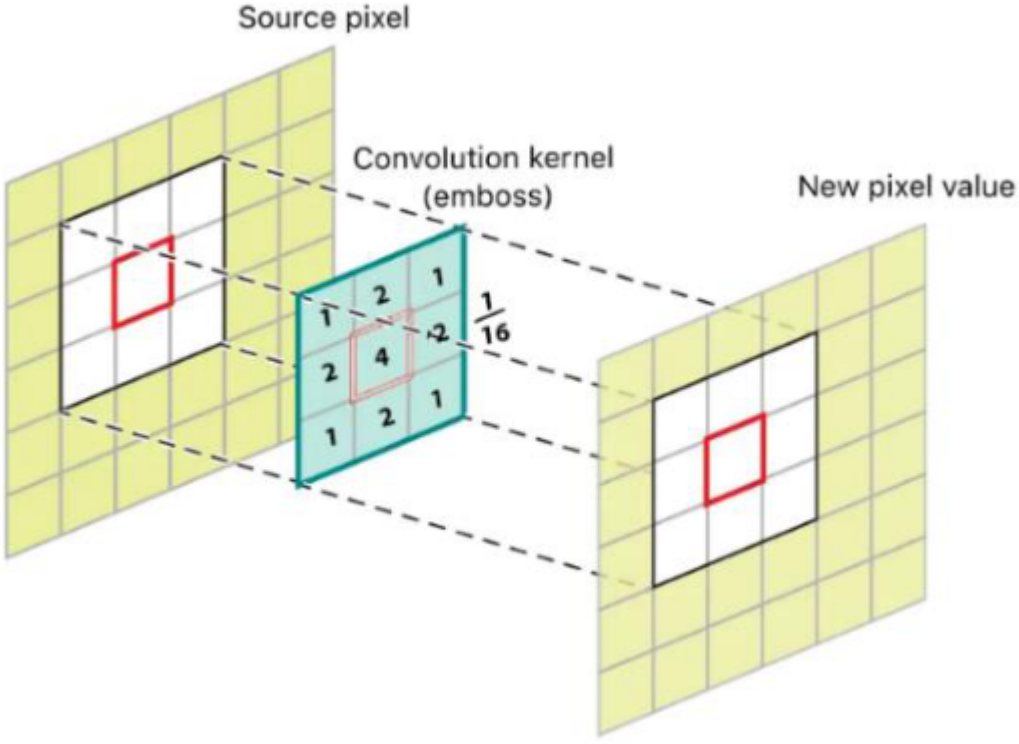
1. **Key Diagrams or Visual Elements**



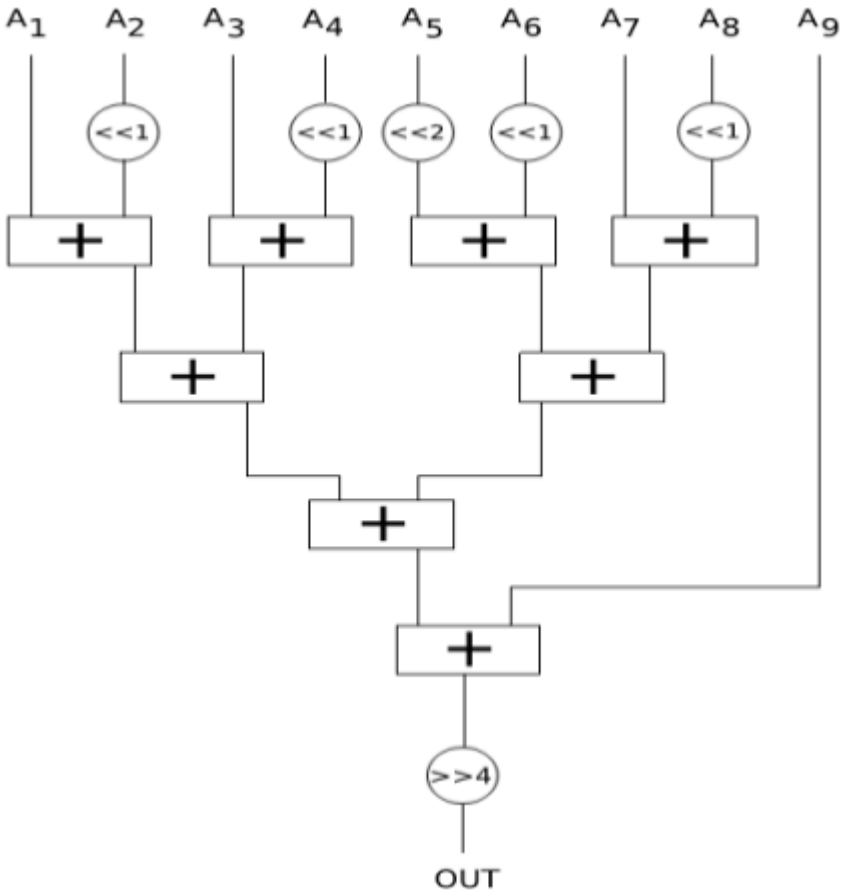
- **Figure 1:** Shows the delay line buffer structure used to store recent pixel values, minimizing memory access during convolution. This is crucial for efficient FPGA implementation.



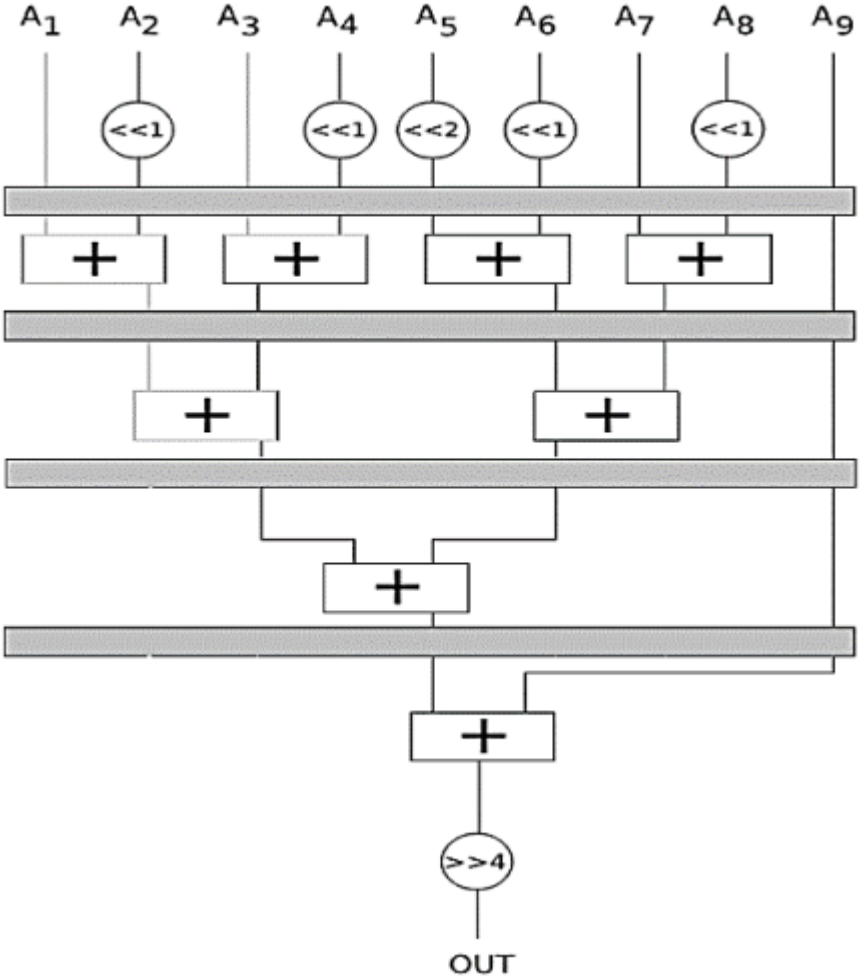
- **Figure 2:** Illustrates the convolution operation with a 3×3 Gaussian kernel sliding over the image pixels, explaining how each output pixel is computed.



- **Figure 3:** Displays the weighted 3×3 Gaussian kernel used for smoothing.



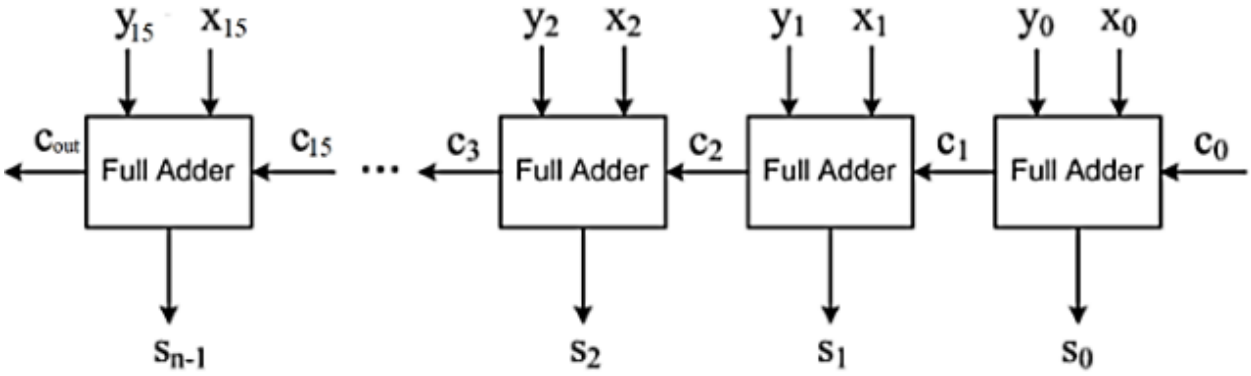
- **Figure 4:** Block diagram of the Gaussian filter showing adders and shifters used for multiplication and division by powers of two (implemented as bit shifts).



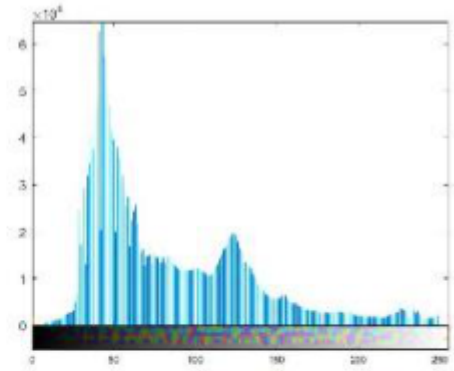
- **Figure 5:** Depicts the pipeline structure dividing the Gaussian filter computation into

four stages to increase processing speed.

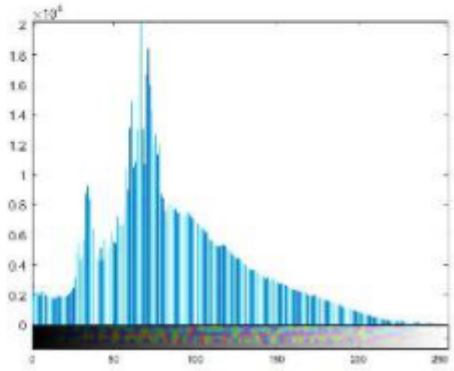
- **Table 1:** Lists logic functions of different approximate full adders (APFAs), showing how sum and carry outputs are simplified to reduce hardware complexity.



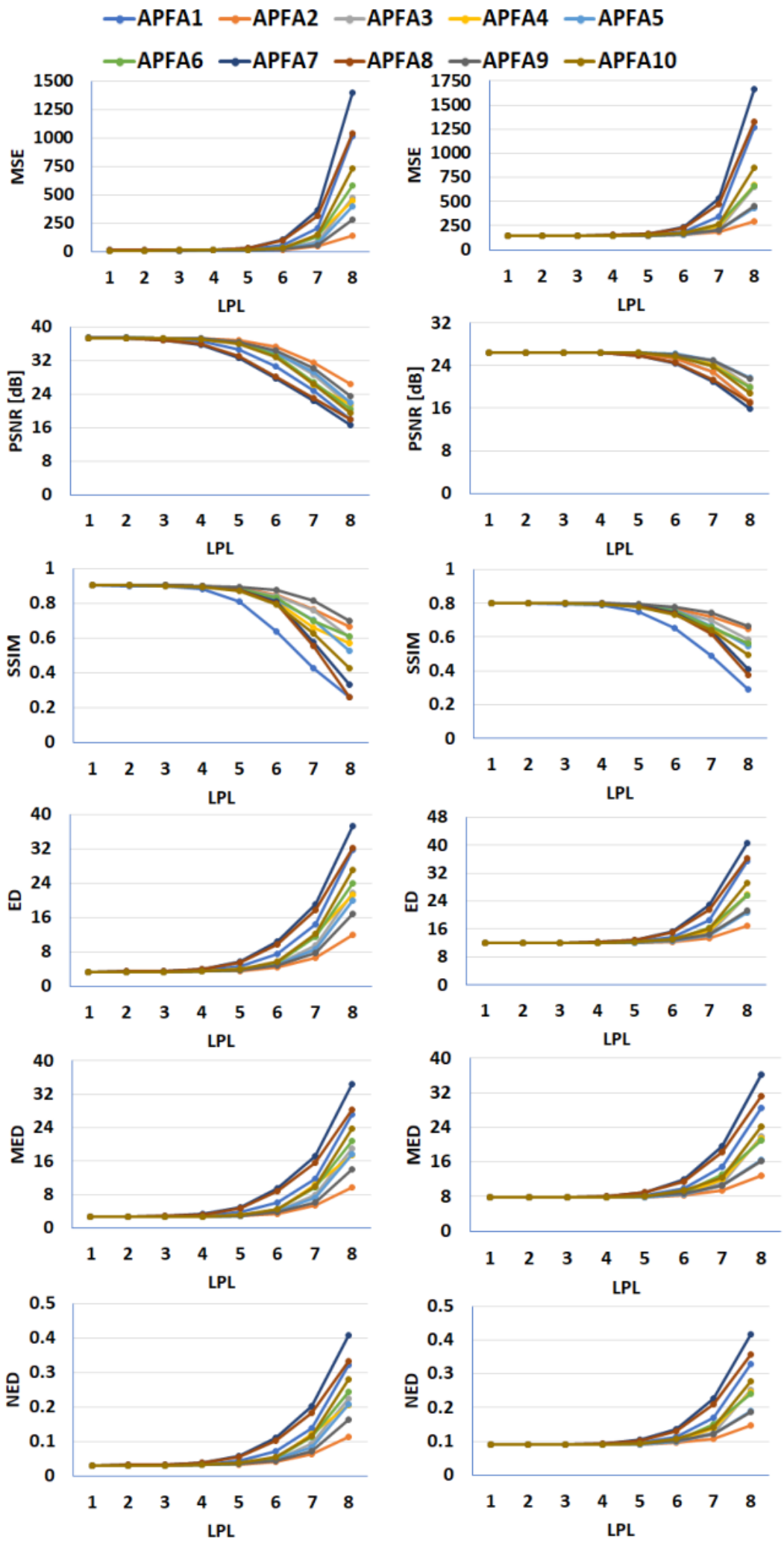
- **Figure 6:** Shows the 16-bit carry ripple adder structure where approximation is applied to least significant bits.



(a)



- **Figure 7:** Presents two sample raw underwater images used for testing, highlighting real-world conditions.


APFA1
APFA2
APFA3
APFA4
APFA5
APFA6
APFA7
APFA8
APFA9
APFA10
MSE
PSNR [dB]
SSIM
ED
MED
NED
LPL