Summary of Research Paper

Underwater Image Enhancement Using FPGA-Based Gaussian Filters with Approximation Techniques
Summary Date: August 2, 2025

1. **Introduction / Abstract**
   The paper addresses the problem of poor visual quality in underwater images caused by natural phenomena 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** (Field-Programmable Gate Array, a type of reconfigurable hardware that allows 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 main findings show that this approach speeds up processing by over 150% and reduces power consumption by more than 34%, though it requires more hardware area. This trade-off is suitable for error-tolerant applications like image and video processing.

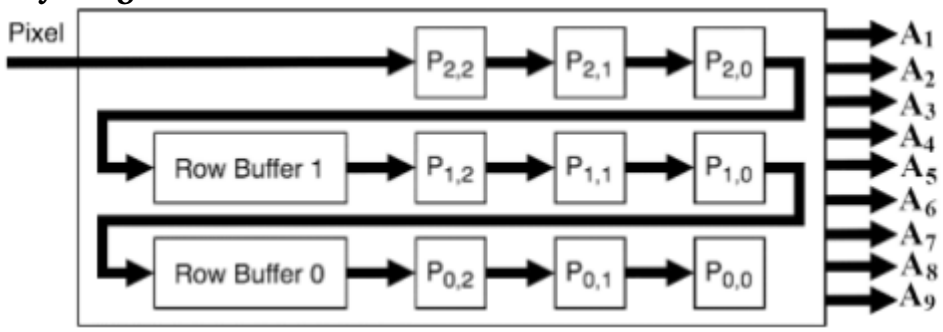
2. **Methodology**
   The researchers designed a pipeline Gaussian filter architecture on an FPGA, which processes image pixels in stages to increase speed. They used line buffers to store image rows and window buffers to hold local pixel areas for filtering, minimizing memory access. The Gaussian filter applies a weighted convolution mask (a 3×3 kernel) to smooth images and reduce noise. To further improve efficiency, they incorporated various approximate adders that simplify addition operations by allowing minor errors, reducing logic complexity, power, and delay. They tested ten different approximate adder designs, applying approximations to the least significant bits of 16-bit adders, and evaluated the impact on image quality and hardware performance using simulations and FPGA synthesis.

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 ((x,y)) are pixel coordinates and (\sigma) controls the spread of the smoothing effect. This function creates a bell-shaped curve that emphasizes central pixels and suppresses distant ones, effectively blurring noise while preserving image structure. The filter is separable, meaning a 2D convolution can be done as two 1D convolutions, improving computational efficiency. The convolution operation combines pixel values with kernel weights to produce a smoothed output pixel.

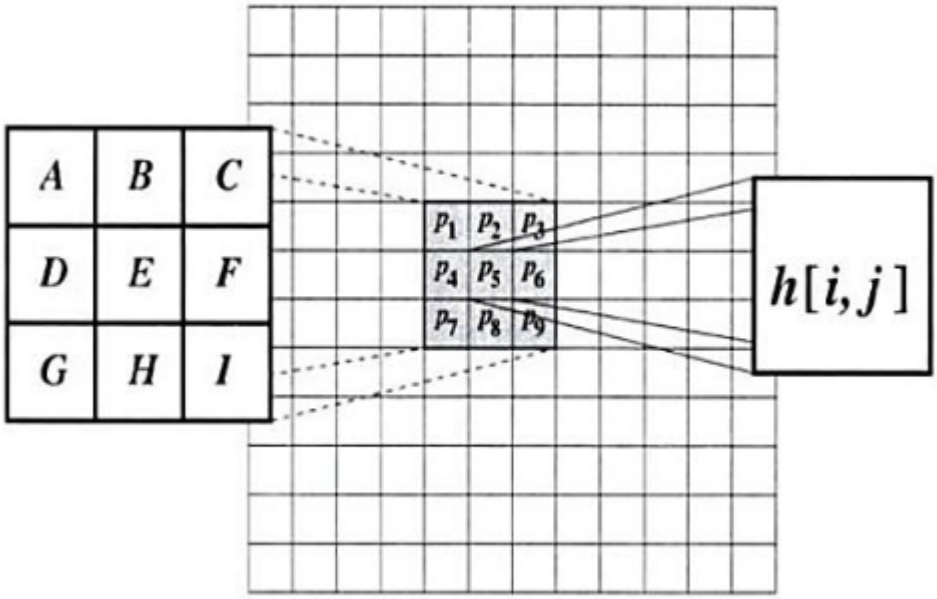
For image quality assessment, the paper uses:

- **PSNR (Peak Signal-to-Noise Ratio):** Measures the ratio of signal power to noise power; higher values indicate better quality.
- **MSE (Mean Square Error):** Average squared difference between original and processed images; lower values are better.
- **SSIM (Structural Similarity Index):** Evaluates perceptual similarity considering luminance, contrast, and structure; values closer to 1 indicate higher similarity.
- **Error Distance (ED), Mean Error Distance (MED), Normalized Error Distance (NED):** Metrics that quantify spatial differences between original and processed images.

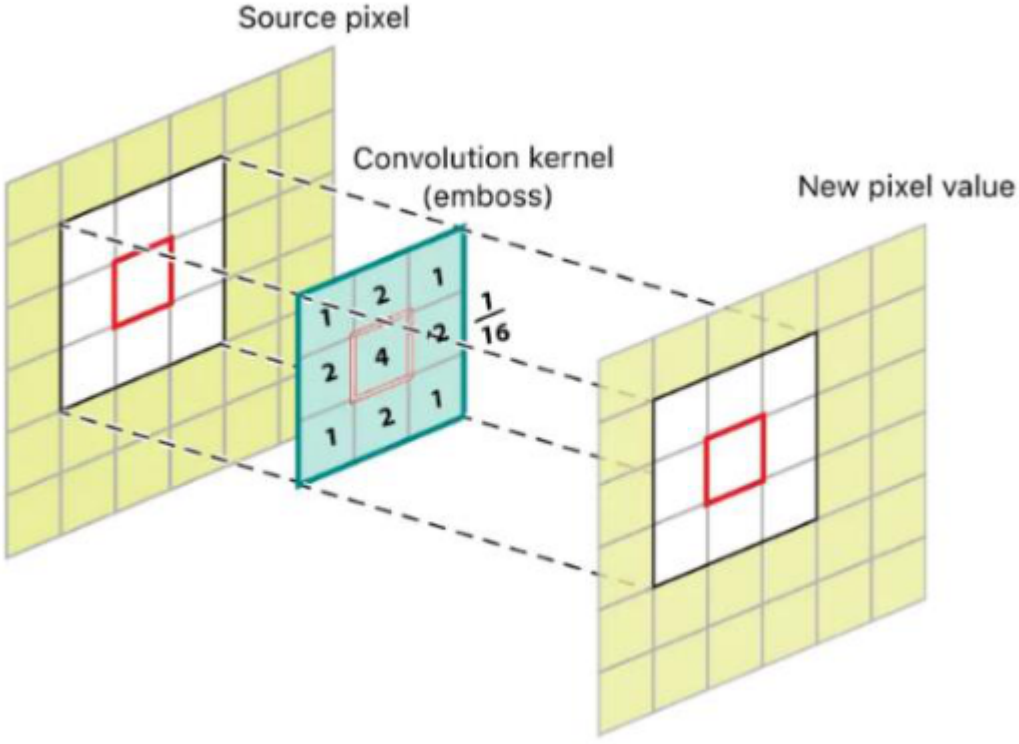
1. **Key Diagrams or Visual Elements**



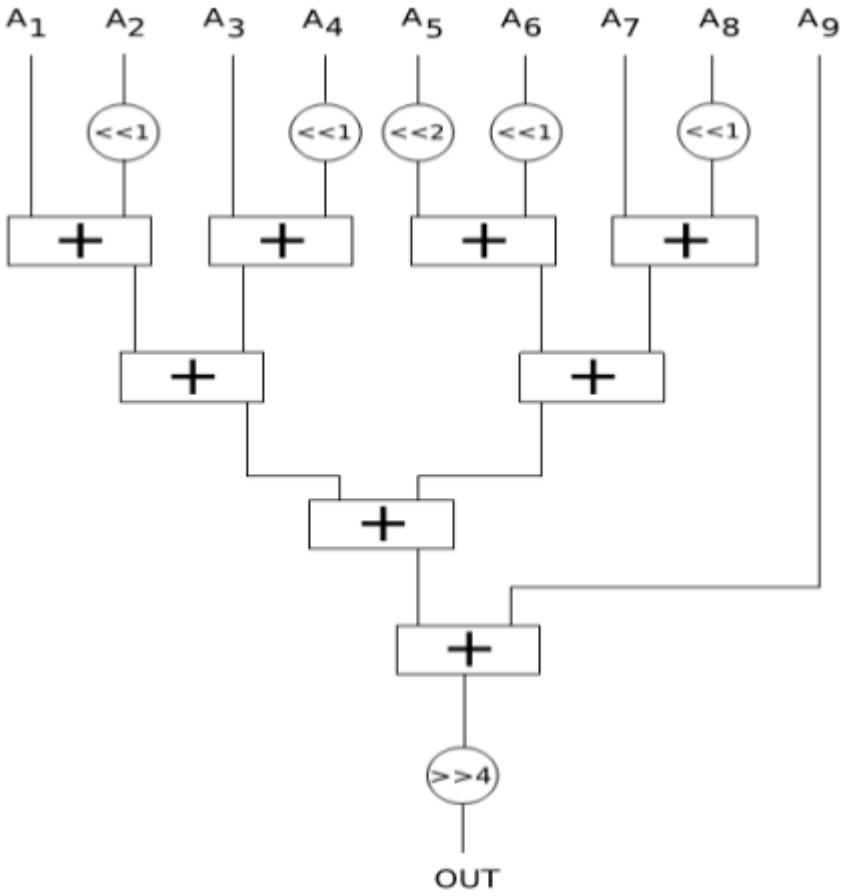
- **Figure 1:** Shows the delay line buffer structure used to store recent pixel values for efficient convolution, minimizing memory access during filtering.



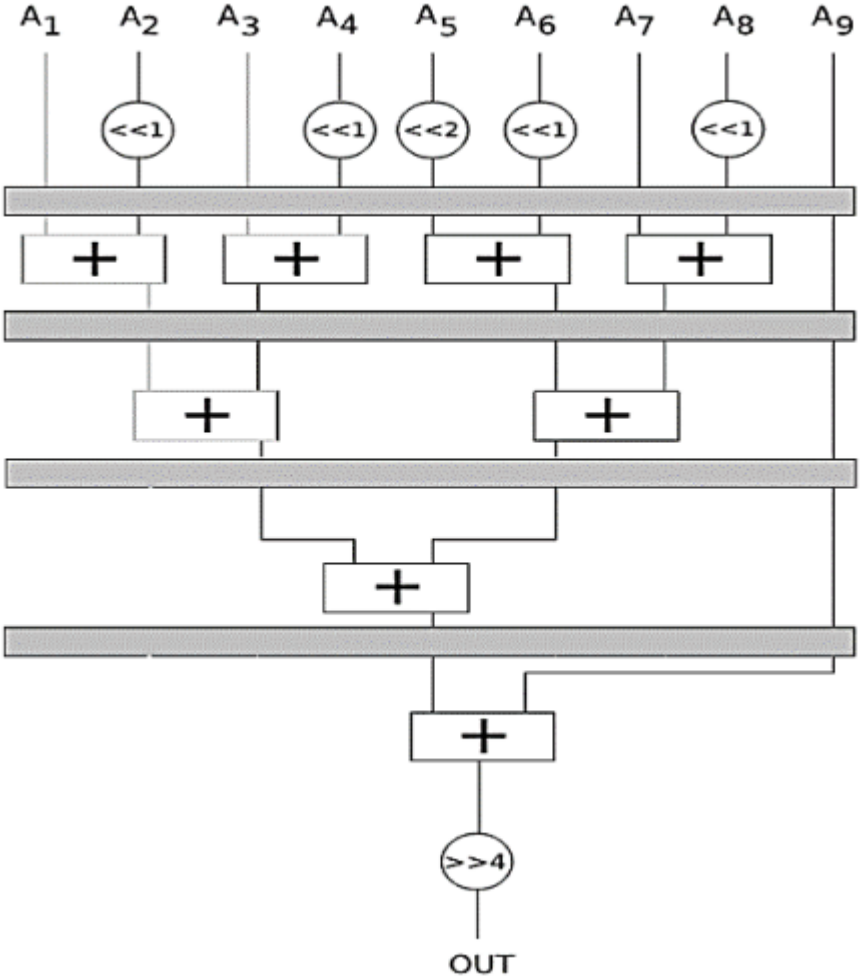
- **Figure 2:** Illustrates the convolution operation with a 3×3 Gaussian kernel sliding over the image pixels, showing how each output pixel is computed from neighboring pixels.



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

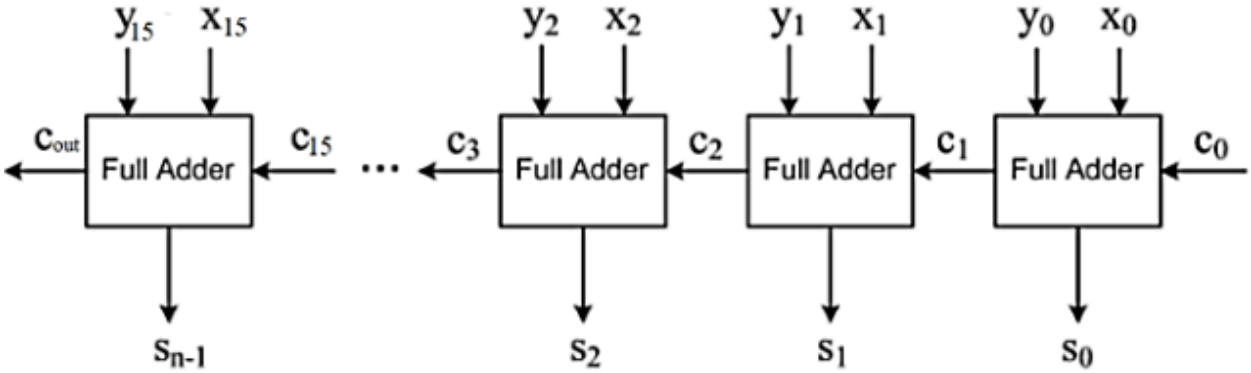


- **Figure 4:** Block diagram of the Gaussian filter implementation, highlighting the use of adders and shifters for multiplication and division by powers of two.

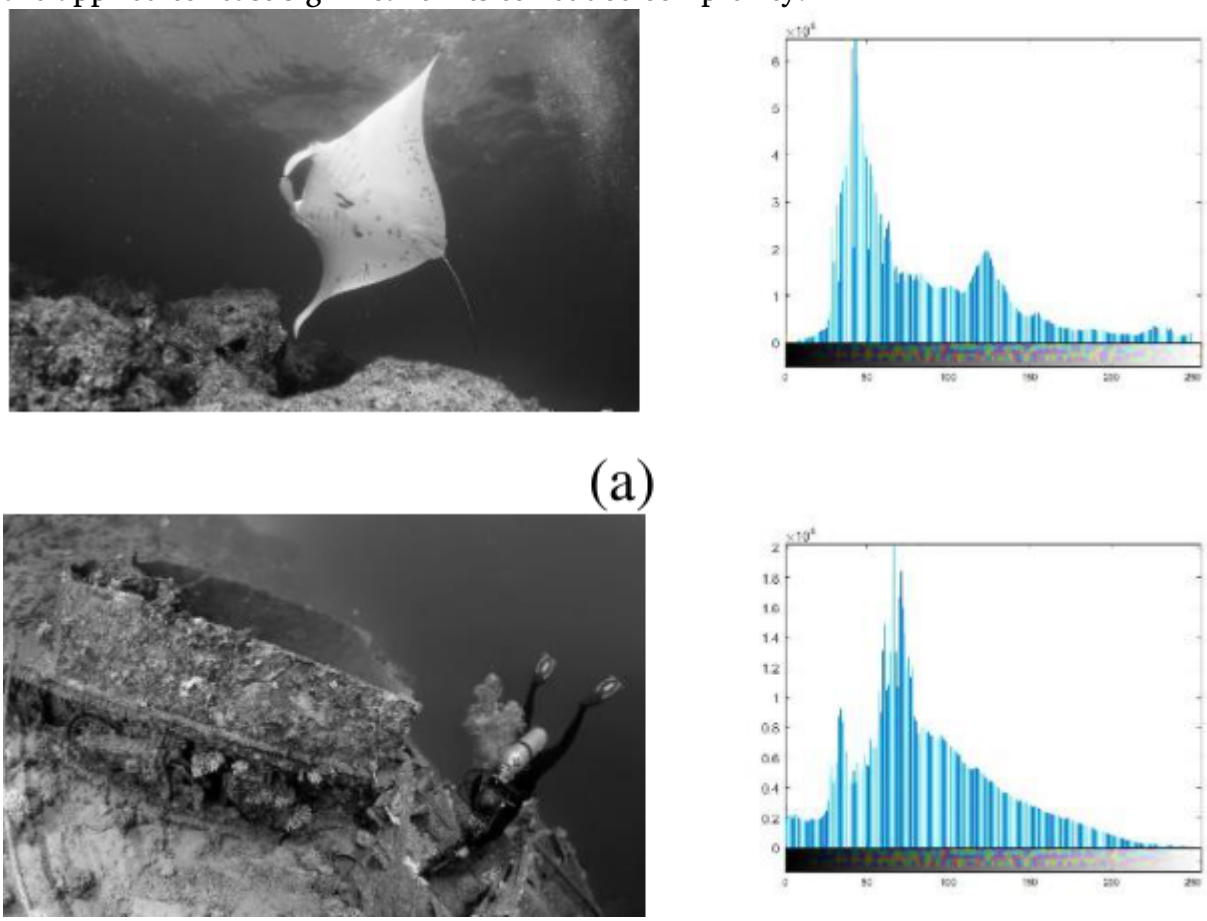


- **Figure 5:** Depicts the pipeline structure of the Gaussian filter, dividing computation

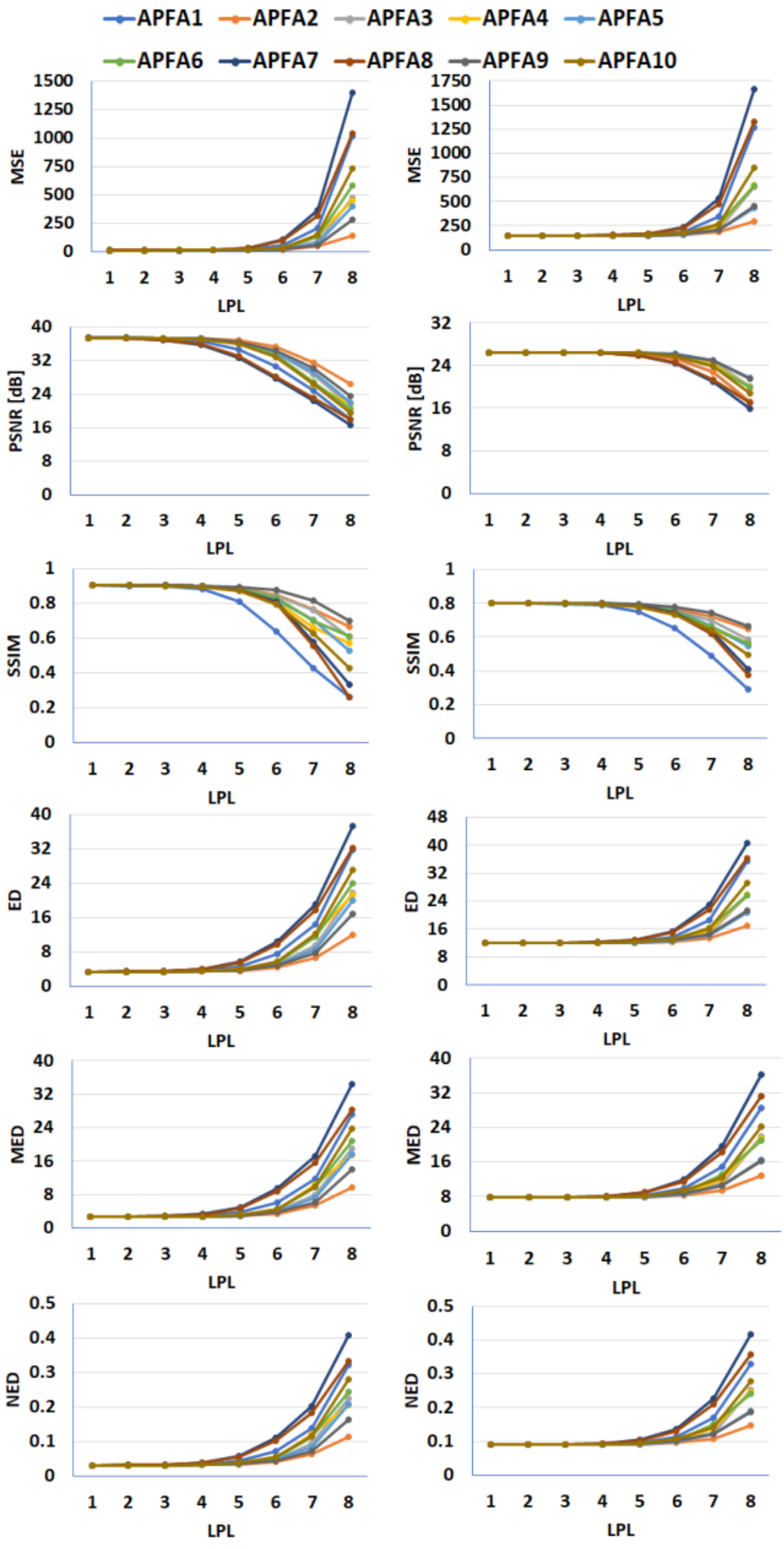
into four stages to enable simultaneous processing and increase speed.



- **Figure 6:** Shows the 16-bit carry ripple adder architecture where approximate adders are applied to least significant bits to reduce complexity.



- **Figure 7:** Presents two raw underwater images used for testing, illustrating typical underwater conditions with noise and color distortion.


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