ADAPTIVE FILTER FUSION AND OPTIMIZED HARDWARE DESIGN FOR IMAGE DENOISING

A Thesis Submitted
in Partial Fulfillment of the Requirements for the
Degree of

MASTER OF TECHNOLOGY

in

Signal Processing and Digital Design

by

MOHIT GARG
(Roll No. 2K23/SPD/11)

Under the Supervision of

Dr. Ajai Kumar Gautam
Associate Professor, ECE, DTU

Department of Electronics and Communication Engineering

DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Shahbad Daulatpur, Main Bawana Road, Delhi-110042. India

May, 2025

ACKNOWLEDGEMENTS

I want to intimate my heartfelt thanks to my project guide, Dr. Ajai Kumar Gautam, Associate Professor and Prof. O P Verma, Head of Department of Electronics and Communication Engineering of Delhi Technological University, for his tremendous support and assistance base on their knowledge. I am so grateful to them for assisting me with the all the necessary tools for the completion of the project. I also want to extend my heartfelt gratitude to all those who have supported my research on Adaptive Filter Fusion and Optimized Hardware Design for Image Denoising. I am grateful to the open-source community for developing and maintaining user friendly deep learning frameworks for simplifying the implementation of the research. I specially feel very thankful for our parents, friends, and classmates for their support throughout my project period. Finally, I express my gratitude to everyone for supporting me directly or indirectly in completing this project successfully. Your support and inspiration have been truly invaluable, which encourages me.

Mohit Garg

(2K23/SPD/11)

DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Shahbad Daulatpur, Main Bawana Road, Delhi-42

CANDIDATE'S DECLARATION

I Mohit Garg, hereby certify that the work which is being presented in the thesis entitled "Adaptive Filter Fusion and Optimized Hardware Design for Image Denoising" in partial fulfillment of the requirements for the award of the Degree of Master of Technology, submitted in the Department of Electronics and Communication Engineering, Delhi Technological University is an authentic record of my own work carried out during the period from August 2023 to May 2025 under the supervision of Dr. Ajai Kumar Gautam, Associate Professor of Department of Electronics and Communication Engineering, Delhi Technological University.

The matter presented in the thesis has not been submitted by me for the award of any other degree of this or any other Institute.

Candidate's Signature

DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Shahbad Daulatpur, Main Bawana Road, Delhi-42

CERTIFICATE BY THE SUPERVISOR(s)

Certified that **Mohit Garg (**2K23/SPD/11) has carried out their search work presented in this thesis entitled **"Adaptive Filter Fusion and Optimized Hardware Design for Image Denoising"** for the award of **Master of Technology** from Department of Electronics and Communication Engineering, Delhi Technological University, Delhi, under our supervision. The thesis embodies results of original work, and studies are carried out by the student herself and the contents of the thesis do not form the basis for the award of any other degree to the candidate or to anybody else from this or any other University/Institution.

Signature

(Dr. Ajai Kumar Gautam)

(Associate Professor)

(DTU, Shahbad Daulatpur,
Main Bawana Road,
Delhi-42)

Date:

ABSTRACT

Image denoising is vital for retaining visual information for tasks such as object detection, classification, and image enhancement. The practical issue with image denoising is one of balancing noise reduction with maintaining key image features, such as edges and textures. The contributions of this thesis are outlined in twin aspects, being a software-based adaptive filter fusion solution of color image denoising, and an optimized hardware implementation of an adaptive filter that performs real-time denoising of gray-scale images. More specifically, this research focused on algorithmic adaptability and hardware efficiency in various noise types such as Gaussian and salt-and-pepper noise.

The first part of the thesis presents an adaptive fusion filtering framework for color image denoising under Gaussian noise for standard deviations of $\sigma = 10$ through $\sigma = 50$. The proposed framework employs a dynamic per-pixel fusion of three carefully selected adaptive filters: Least Mean Squares (LMS), Llncosh, and VSLMS Ang's. These filters were chosen as they complement each other based on their strengths in texture sensitivity, edge preservation, and stability following benchmarking studies where we analyzed the performance. The merger utilizes an inverse squared error−based weighting scheme that spatially adapts as a function of the image pixel locations under denoising. The use of bilateral preprocessing enables improved edge-aware smoothing characteristics while retaining fine-level structural details. The adaptive filter fusion strategy improves both PSNR and SSIM image quality metrics significantly over using the individual filters indicated above, as shown through the testing on the CBSD68 dataset. Bias-variance analysis and run time profiling support the purpose and practicality of the proposed scheme.

The second component examines the hardware modeling of a fixed-weight Least Mean Fourth (LMF) filter where denoising of grayscale images occurs in an impulsive noise environment. The Least Mean Fourth (LMF) filter has been selected as the approach is insensitive to outliers via the approach to adjusting the fourth power of the error. The LMF filter is trained off-line utilising grayscale images from the BSDS500 dataset, which features artificial added salt-and-pepper noise to emulate images with impulsive corruption. The adaptation employs fixed point arithmetic to provide for field programmable gate array (FPGA) deployment. A standard 3×3 convolution is employed with a streaming architecture to avoid buffering the entire frame and allows for line buffer-based processing. Wallace tree multipliers were used to minimize log delay in the datapath and allow synthesis timing closure without altering the output, and to accelerate the overall implementation. The final architecture achieves a greater than 90% reduction in resource utilization compared to the non-optimized multiplier-based design, yet with nearly identical output fidelity to the software reference.

LIST OF PUBLICATIONS

[1] Mohit Garg and Ajai Kumar Gautam, “Fusion-Based Adaptive Filtering for Color Image Denoising with Dynamic Error Weighting”. [**Accepted**]

[2] Mohit Garg and Ajai Kumar Gautam, “Wallace Tree based Efficient Hardware Modeling of Streaming LMF Filter for Real-Time Image Denoising”.
[**Accepted**]

TABLE OF CONTENTS

Title **Page No.**

Title **Page No.**

LIST OF TABLES

LIST OF FIGURES

Title **Page No.**

LIST OF ABBREVIATIONS

Abbreviation	Full Form
LMF	Least Mean Fourth
LMS	Least Mean Squares
VSLMS_Ang	Variable Step-Size LMS (Angular variant)
RLS	Recursive Least Squares
NLMS	Normalized Least Mean Squares
Llncosh	Log-Cosh Adaptive Filter
PSNR	Peak Signal-to-Noise Ratio
SSIM	Structural Similarity Index
MAC	Multiply-Accumulate
HDL	Hardware Description Language
RTL	Register Transfer Level
FPGA	Field-Programmable Gate Array
LUT	Look-Up Table
FF	Flip-Flop
DSP	Digital Signal Processing (Block)
BRAM	Block RAM
NLM	Non-Local Means
BM3D	Block Matching and 3D Filtering
YCbCr	Luminance-Chrominance Color Space
RGB	Red Green Blue
DnCNN	Denoising Convolutional Neural Network
AWGN	Additive White Gaussian Noise
I/O	Input/Output
σ (sigma)	Standard Deviation (Noise Level)
CBSD68	Color Berkeley Segmentation Dataset (68 Images)
BSDS500	Berkeley Segmentation Dataset (500 Images)
Vivado	Xilinx Vivado Design Suite

CHAPTER 1

INTRODUCTION

1.1 Importance of Image Denoising in Vision and Imaging Systems

Image denoising is a core process in digital image processing and computer vision. Its essential purpose is to enhance the quality of images by removing noise while retaining useful visual information characteristics. Clean images are fundamental to the precise implementation of higher-level tasks (e.g., object detection, classification, segmentation, or feature extraction) in medical imaging, remote sensing, autonomous navigation, and monitoring applications [1], [2].

In the real-world applications, image acquisition systems are regularly exposed to noise as a result of the hardware sensors limitations, adverse environmental conditions, and signal transmission imperfections. All of these introduce random variations in pixel intensities and can result in considerably low-quality images and unreliable downstream processing. Thus, it is essential to develop robust and efficient denoising algorithms for preserving visual data and ensuring its usability across both software applications and hardware-constrained settings [3], [4].

1.2 Challenges of Gaussian and Salt-and-Pepper Noise

Two common types of noise that impact digital images are additive Gaussian noise and salt-and-pepper (also called impulse) noise. Gaussian noise is due to thermal fluctuation in the sensor, electronic interference (which can be external or internal), and quantization errors. It is usually modeled as zero-mean white noise at all intensities. The presence of Gaussian noise in an image can cause loss of detail in the image, decrease the contrast of the image, and can blur high-frequency information such as edges and textural information [5, 7].

Salt-and-pepper noise, on the other hand, is presented as randomly occurring black (0) and white (255) pixels in the image due to different causes, generally due to sensor element malfunction, bit error in transmission or faults in analog-to-digital conversion [8], [9]. In contrast to Gaussian noise, impulsive noise generates sharp intensity spikes that drastically distort local pixel information and complicate the ability to effectively apply linear smoothing techniques. Both types of noise have specific demands and challenges. Specifically, Gaussian noise requires retaining the structure of the image but suppressing the smooth deviation and salt-and-pepper noise requires outlier robustness that maintains neighboring image features.

On top of this, color images become more complicated given the dependence of the RGB channels. Noise in any channel can impact perceived quality across the whole color image, which calls for designing denoising methods that are both color-aware and structure-aware [6], [10].

1.3 Limitations of Individual Filters and Traditional Denoising

Conventional denoising approaches, such as median filtering, bilateral filtering, and wavelet shrinkage, have long been used due to their simplicity and ability to handle low- to moderate-noise scenarios [11], [12]. Median filters are particularly effective against salt-and-pepper noise, while bilateral filters provide a good trade-off between noise suppression and edge preservation [4], [13]. However, these methods rely on fixed spatial or statistical models, making them less effective when noise levels are high or image content varies significantly.

Adaptive filters provide a more flexible framework by adjusting their parameters in response to the input signal, as shown in block diagram in Fig. 1.1. Techniques such as the Least Mean Squares (LMS), Recursive Least Squares (RLS), and Least Mean Fourth (LMF) filters dynamically update their weights based on the observed error between predicted and actual pixel values [11], [14]. This allows them to adapt to local variations in the image and respond differently to flat, textured, or edge-rich regions.

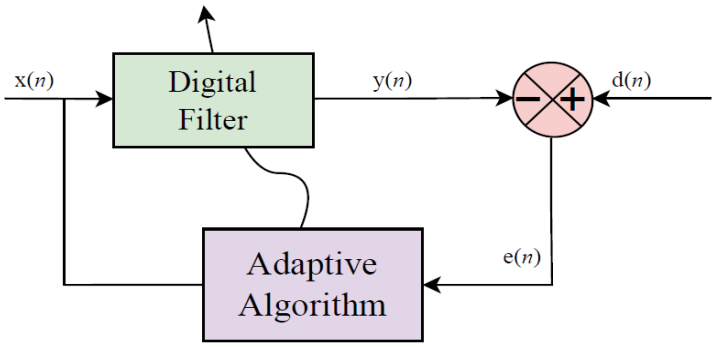


Fig. 1.1 Adaptive Filter Block Diagram

Despite their theoretical advantages, individual adaptive filters face several limitations. Many are sensitive to initialization, require careful tuning of step sizes, and may exhibit instability in the presence of high variance or non-stationary noise [15], [16]. Moreover, a single filter is unlikely to perform optimally across the entire image, especially in color images where different regions demand different filtering characteristics. As a result, the effectiveness of any standalone filter is inherently constrained, prompting the need for hybrid or fusion approaches [17], [18].

1.4 Motivation for Adaptive Filter Fusion and Hardware Optimization

The limitations of traditional and standalone adaptive filters motivate the development of a fusion-based adaptive filtering framework. The central idea is to combine the strengths of multiple adaptive filters—each having distinct error characteristics—into a unified system that dynamically adjusts its behavior based on pixel-wise performance. For example, one filter may be more effective in flat regions while another excels at preserving edges or textures. By assigning greater weight to the best-performing filter at each pixel location, the fused output can achieve better overall denoising quality than any individual filter [6], [19].

In this work, a fusion model is proposed based on inverse squared error weighting of three adaptive filters: LMS, Llncosh, and VSLMS Ang's. The fusion weights are computed in real time based on local error metrics, enabling the system to respond intelligently to different regions of the image. Additionally, a bilateral preprocessing stage is incorporated to improve edge definition and enhance the quality of input patches, especially under Gaussian noise [20], [21].

On the hardware front, the need for real-time image denoising necessitates efficient architectures that can process data with minimal latency and resource consumption. The LMF filter, known for its robustness to salt-and-pepper noise due to its fourth-power error cost function, is chosen for hardware implementation [9], [22]. However, its computational demands are addressed through architectural optimizations. Specifically, the use of Wallace tree multipliers in the filtering path significantly reduces the critical path delay, logic utilization, and power consumption [23], [24]. The resulting design is capable of streaming input data and producing filtered output at high throughput—making it well-suited for FPGA-based or embedded deployment [25], [26].

1.5 Summary of Contributions

1.5.1 Adaptive Filter Fusion with Bilateral Preprocessing

A fusion-based framework is developed that combines three distinct adaptive filters—LMS, Llncosh, and VSLMS Ang's—using a dynamic, pixel-wise inverse error weighting scheme. Bilateral filtering is employed as a preprocessing step to enhance structural fidelity, especially in edge and texture-rich areas. A bias-variance analysis is conducted to theoretically and empirically validate the fusion performance. The fused output is shown to exhibit reduced variance compared to individual filters [16], [18].

The computational efficiency of the proposed approach is evaluated against popular classical methods (NLM, BM3D, DnCNN), demonstrating that the fusion method achieves competitive denoising performance while requiring significantly fewer multiply-accumulate operations (MACs).

1.5.2 Wallace Tree–Optimized Real-Time Hardware Filter

A Verilog-based LMF filter is implemented using fixed weights derived from offline adaptive training. The hardware architecture employs a streaming line-buffer design with 3×3 window processing, including valid pixel detection and border skipping. To optimize the arithmetic pipeline, Wallace tree multipliers replace conventional multipliers, resulting in notable reductions in logic utilization and dynamic power, with improved timing closure. The hardware-filtered output closely matches the software reference output, confirming functional equivalence with improved throughput and resource efficiency [23], [24].

1.6 Thesis Organization

Chapter 2: Literature Survey
Reviews conventional denoising techniques, adaptive filters, fusion approaches, and prior hardware implementations of image filters.

Chapter 3: Methodology 1
Describes the adaptive fusion framework, including dataset, filter selection, weighting mechanism, and bilateral preprocessing strategy.

Chapter 4: Methodology 2
Presents the Verilog hardware design of the LMF filter, including system architecture, streaming logic, and Wallace tree optimization.

Chapter 5: Experimental Analysis
Details the evaluation of both software and hardware implementations using PSNR, SSIM, complexity, and resource metrics.

Chapter 6: Final Analysis and Future Scope
Discusses the observed results, draws comparisons with existing techniques, concludes the work, and proposes directions for future research.

CHAPTER 2

LITERATURE SURVEY

2.1 Classical Image Denoising Approaches

Early image denoising research centered around the suppression of additive noise using spatial and transform-domain filters. Traditional filters such as the mean and median were simple but often compromised edge and texture information. A major breakthrough came with the Non-Local Means (NLM) algorithm by Buades et al., which introduced the use of distant self-similar patches for weighted averaging. This preserved structural details significantly better than classical local filters, especially in flat regions with repetitive patterns [1].

Building on this, Block-Matching and 3D Filtering (BM3D) by Dabov et al. became a gold standard for Gaussian noise removal, delivering state-of-the-art results through collaborative filtering in grouped 3D transform coefficients [3]. However, both NLM and BM3D were computationally expensive and unsuitable for real-time use.

Wavelet thresholding and Wiener filtering were also explored for denoising in transform domains, leveraging signal sparsity. Techniques like VisuShrink and BayesShrink removed noise in wavelet coefficients but underperformed in textured or color-rich images [2].

2.2 Progress in Edge-Preserving Filtering

With the need to preserve structure, bilateral filtering emerged as a robust approach that combined spatial and radiometric proximity for weighted smoothing. Though effective in edge preservation, it was costly in large images [4]. Guided filtering offered a fast alternative with linear complexity, maintaining edge-awareness using a reference guidance image [5]. For color image denoising, where inter-channel dependencies matter, Dinh et al. proposed cross-channel texture transferring to retain chromatic consistency [6], while quaternion-based methods unified RGB modeling to reduce color artifacts [7].

These methods advanced color fidelity but still lacked adaptive flexibility in textured or dynamic regions.

2.3 Adaptive Filtering: LMS, LMF, and Beyond

Adaptive filtering introduced parameter updates via feedback from input-output errors. The LMS algorithm, popular for its simplicity, updated weights using gradient descent and was widely used in digital signal processing [19]. However, LMS was sensitive to non-Gaussian noise and outliers.

To address impulsive noise, the Least Mean Fourth (LMF) algorithm was introduced, which penalized large errors more heavily by minimizing the fourth power of the error. This made LMF effective in salt-and-pepper and impulsive environments [14]. However, it came with slower convergence and higher computation.

Improved variants like Normalized LMF (NLMF) and bias-compensated LMF achieved faster convergence and robustness under dynamic conditions [8], [11]. The Llncosh filter, using a log-likelihood-based cost, offered better generalization and smooth adaptation in Gaussian noise settings [27]. Regularization techniques further enhanced adaptive filter stability under continuous updates [16].

2.4 Fusion-Based Strategies

Despite individual strengths, adaptive filters are prone to local overfitting and instability. Inspired by fusion learning, fusion-based methods combine outputs from multiple filters, each optimized for different local conditions [18]. Fusion assigns adaptive weights using performance indicators like absolute or squared error.

A notable approach is inverse squared error weighting, where each filter's output is weighted inversely proportional to its error, allowing more accurate filters to dominate locally [15]. Region-based switching and low-rank fusion are other strategies that leverage structural correlations across filters [9], [17].

In color image denoising, bilateral preprocessing enhances spatial consistency, helping preserve edge transitions and reducing noise gradients prior to fusion. This results in improved robustness across RGB channels [28].

Empirical evaluations consistently show fusion-based models outperform individual filters in PSNR, SSIM, and perceptual metrics across multiple noise levels [20], [21].

2.5 Deep Learning for Image Denoising

Deep learning models revolutionized denoising by learning noise mappings from data. DnCNN employed residual learning, training CNNs to predict and subtract noise components, achieving remarkable performance on Gaussian noise [13]. Later models like FFDNet introduced variable noise-level conditioning, while DRUNet integrated attention and dilation for improved performance [26], [29].

Though highly effective, these models are computation-intensive, data-dependent, and less interpretable than traditional adaptive filters—limiting their real-time hardware deployment potential.

2.6 Hardware-Oriented Adaptive Filtering

For embedded use, LMS filters have been synthesized on FPGA and ASIC platforms using fixed-point arithmetic for area and power efficiency [24], [25]. Multiplier-less LMS and distributed arithmetic LMS eliminated costly hardware multipliers with logic-optimized alternatives [12], [22].

Though computationally heavier, the LMF filter remains attractive due to its superior performance under impulsive noise. The main challenge lies in its reliance on high-speed multiplication operations.

To address this, Wallace tree multipliers are adopted. Originally proposed by Wallace, these reduce adder stages and critical path delay in binary multiplication arrays [30]. Subsequent VLSI- and FPGA-optimized versions of Wallace trees improved speed and throughput significantly [31].

Comparative analysis has shown Wallace tree multipliers consistently outperform Booth and array multipliers in area-delay tradeoffs and pipeline efficiency, particularly for 24-bit and higher operations, making them ideal for LMF-based filtering hardware [10].

2.7 Summary and Positioning

In summary, denoising methods have evolved from spatial filtering to adaptive and fusion strategies, culminating in real-time hardware-accelerated solutions. Each generation addressed limitations in structure preservation, computational complexity, and noise generalization. This thesis contributes to this lineage through:

- A fusion-based adaptive filter framework using LMS, Llncosh, and VSLMS with bilateral preprocessing and inverse-error weighting.
- A streaming hardware-optimized LMF filter accelerated with Wallace tree multiplication for real-time image denoising.

Together, they form a practical and scalable approach to denoising that balances statistical accuracy, perceptual quality, and implementation feasibility.

CHAPTER 3

Methodology 1: Adaptive Filter Fusion for Color Image Denoising

3.1 Dataset Description

To evaluate the performance of the proposed adaptive filter fusion framework, we utilize the CBSD68 dataset, a widely adopted benchmark in image denoising research [1]. The dataset comprises 68 natural color images with diverse textures, edges, and luminance characteristics. This diversity makes it ideal for assessing the spatial adaptability and generalization capacity of denoising algorithms.

For experimental consistency, additive white Gaussian noise is synthetically applied to each image. Four noise levels—standard deviations of $\sigma = 10$, 15, 25, and 50—are used to simulate different noise intensities. These levels reflect realistic conditions ranging from light sensor noise to severely degraded transmission scenarios. The clean, uncorrupted images are retained as ground truth for computing full-reference quality metrics such as PSNR and SSIM [3].

3.2 Filter Selection and Benchmarking

To construct a high-performing and efficient adaptive fusion framework, a variety of adaptive filters were evaluated under Gaussian noise with $\sigma = 15$, a representative mid-level noise condition. The benchmarking focused on denoising quality (measured using PSNR), using the CBSD68 dataset with full-color inputs. Results of the Benchmarking are shown in Table 3.1.

Table 3.1 Adaptive Filter PSNR Ranking

Rank	Filter	Average PSNR (dB)	Notes
1	RLS	29.71	Best quality; very high computational load.
2	VSLMS Ang's	29.43	Directionally adaptive; edge-aware learning.
3	Llncosh	29.32	Smooth and stable convergence.
4	LMS	28.96	Simple, effective in smooth regions.

3.2.1 Justification for Filter Selection

While RLS delivered the highest PSNR, it was excluded due to its significant computational overhead. Specifically:

- RLS performs recursive matrix updates, leading to quadratic time complexity per iteration and elevated memory usage.
- In practice, RLS exhibited very high runtime than LMS or Llncosh in Python (see Table 5.2, runtime analysis).
- This renders it unsuitable for either dynamic fusion (per-pixel weighting) or hardware implementation [14].

Thus, three filters were selected based on their strong trade-off between denoising performance, computational cost, and complementary behavior:

- LMS: Efficient in smooth/flat regions; fast convergence
- Llncosh: Edge-preserving; well-behaved near gradients
- VSLMS Ang's: Excels in high-texture, detail-preserving regions

Together, these filters span a diverse adaptation space and are computationally efficient enough to be used for per-pixel dynamic fusion, a key aspect of the proposed methodology.

3.3 Fusion-Based Filtering Architecture

The proposed architecture fuses the outputs of the three selected adaptive filters on a pixel-wise basis, dynamically adjusting each filter's contribution according to its local estimation accuracy. The framework is designed to process full-color images, with the fusion process applied separately on each RGB channel, thereby maintaining inter-channel consistency and preserving color integrity [6].

The denoising pipeline consists of the following major components:

- Bilateral Preprocessing: Applied to the noisy image to enhance structural continuity and suppress coarse noise
- Adaptive Filtering Stage: The preprocessed image is fed into the three selected adaptive filters operating in parallel
- Dynamic Weighting and Fusion: The output of each filter is combined using a pixel-wise inverse squared error weighting strategy
- Final Output: A weighted summation yields the denoised image

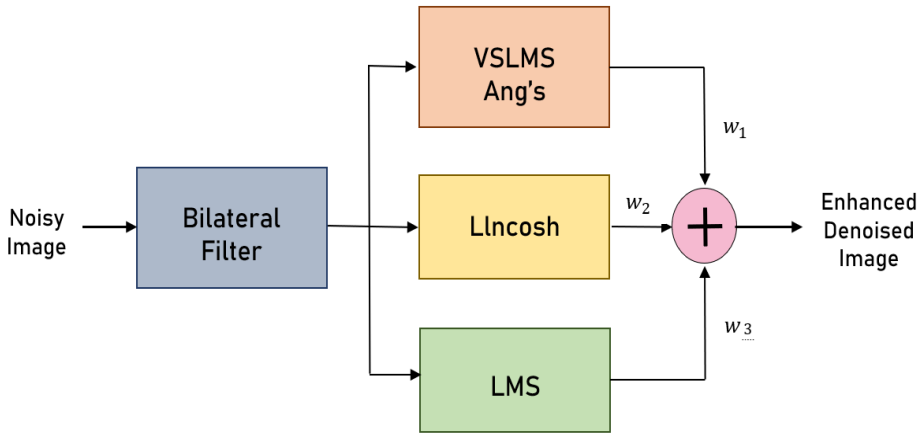


Fig 3.1 Block diagram of proposed fusion-based adaptive filtering system

The block diagram of proposed system is shown in Fig. 3.1. The noisy image is first passed through a bilateral filter to reduce noise while preserving edges. The output is then processed by three adaptive filters—VSLMS Ang's, Llncosh and LMS—in parallel. Their outputs are combined using dynamic error weighting, where each filter's contribution is adjusted based on its local pixel-wise error. The final image is produced by a weighted fusion of these filtered outputs, yielding an enhanced denoised result [7].

3.4 Inverse Squared Error Weighting

The core of the proposed framework is a dynamic error-weighted fusion mechanism that combines the outputs of selected filters based on their instantaneous adaptation error.

At each pixel n, the local error for the i^{th} filter is defined as:

$$e_i(n) = d(n) - y_i(n) \tag{3.1}$$

Where, $y_i(n)$ is the filter output and $d(n)$ is the corresponding ground truth. The contribution of each filter is controlled using inverse squared error weighting:

$$w_i(n) = \frac{1}{e_i(n)^2 + \varepsilon} \tag{3.2}$$

Here, ε is a small constant to avoid division by zero. The weights are then normalized:

$$\widehat{\{w\}_i}(n) = \frac{w_i(n)}{\sum_{j=1}^{K} w_j(n)} \tag{3.3}$$

Finally, the fused output is computed as the weighted sum of all filter outputs:

$$Y_{fused}(n) = \sum_{i=1}^{K} \widehat{\{w\}_i}(n) \cdot y_i(n) \quad (3.4)$$

This fusion mechanism allows the system to adaptively emphasize the best-performing filters at each pixel, thereby improving both perceptual and structural denoising performance [9] , [10].

3.5 Per-Pixel Fusion Behavior

This per-pixel fusion approach leverages the behavioral diversity of the selected filters:

- LMS is generally favored in flat and homogeneous regions due to its simplicity and stability.
- VSLMS Ang's becomes dominant near high-gradient zones such as edges or textures, owing to its directional sensitivity and adaptive learning rate.
- Llncosh tends to balance performance in intermediate regions where local contrast transitions gradually [8].

By dynamically combining these behaviors, the fusion method achieves enhanced denoising performance while minimizing global parameter tuning or segmentation heuristics.

3.6 Bilateral Preprocessing

To further improve filtering quality, bilateral filtering is applied as a preprocessing step prior to adaptive filtering. Bilateral filtering is an edge-preserving smoothing technique that reduces noise while maintaining important structural content. It leverages both spatial proximity and pixel intensity similarity to perform localized smoothing without blurring critical edges [16], [27]. In our pipeline, bilateral filtering improves the quality of filter input patches and enhances the effectiveness of the downstream fusion-based adaptation. This hybrid approach leverages the spatial sensitivity of bilateral filtering and the learning behavior of adaptive filters for robust color image denoising.

3.7 Runtime and Complexity Profiling

In addition to denoising quality, the fusion model is profiled in terms of:

- Multiply-Accumulate Operations (MACs) per image
- Runtime (in seconds) on a standard CPU

The LMS, Llncosh, and VSLMS Ang's filters require only linear or mildly non-linear updates, keeping their computational demands manageable. The overall fusion process—including bilateral filtering and weighted averaging—remains

significantly more efficient than conventional methods like NLM or BM3D [3], [18]. On average:

- The fusion method requires ~12.9 million MACs per 480×321 image
- Typical execution time is < 0.02 seconds per image (CPU-only)

This makes the framework suitable for real-time applications or integration into embedded systems with constrained resources.

3.8 Tools and Implementation

The entire fusion pipeline is implemented in Python using:

- Padasip: For adaptive filtering (LMS, Llncosh, VSLMS Ang's)
- scikit-image (skimage): For image preprocessing, bilateral filtering, and metric evaluation
- NumPy: For vectorized operations and weight normalization
- Matplotlib: For plotting visual results and performance trends

The modular implementation allows for easy replacement or extension of individual filters, facilitating further exploration of alternative fusion strategies.

CHAPTER 4

Methodology 2: Hardware Design and Optimization for LMF Filter

4.1 Overview and Motivation

While adaptive filtering is highly effective for image denoising, software-based implementations are often unsuitable for real-time applications, particularly in embedded or FPGA-based systems. High computational complexity, memory latency, and limited parallelism are major bottlenecks in real-world deployment [20], [26].

To overcome these limitations, we propose a hardware-efficient implementation of the Least Mean Fourth (LMF) adaptive filter using a streaming architecture. The design is tailored for grayscale image denoising under salt-and-pepper noise, a scenario where the LMF filter excels due to its robustness to impulsive disturbances [14]. The design further incorporates Wallace tree multipliers to reduce resource usage and improve processing speed, enabling practical deployment on FPGA platforms [23], [24].

4.2 Dataset and Preprocessing

The hardware model is evaluated using images from the BSDS500 dataset, a benchmark widely used for image segmentation and filtering tasks [17]. To simulate impulsive noise conditions, salt-and-pepper noise is synthetically added using a MATLAB-based preprocessing script.

- Noise Density: 1% (i.e., 1% of total pixels are replaced with either 0 or 255)
- Salt-to-Pepper Ratio: 0.5 (equal probability of black or white pixel corruption)

Images are resized to a fixed resolution of 480 × 321 pixels, and the corrupted noisy images are used as inputs to both software and hardware filters for comparative evaluation.

4.3 Filter Selection for Hardware Implementation

The choice of the Least Mean Fourth (LMF) filter for hardware implementation was driven by both its theoretical robustness in handling impulsive noise and its practical suitability for fixed-weight deployment in real-time systems [11], [26].

4.3.1 Superior Performance in Impulsive Noise

Unlike LMS and RLS variants that minimize squared error, the LMF algorithm minimizes the fourth power of the error signal:

$$J(n) = e(n)^4 \tag{4.1}$$

This higher-order cost function inherently places more penalty on large errors, making LMF particularly effective in suppressing outliers. This property is ideal for denoising salt-and-pepper noise, which introduces extreme pixel deviations (i.e., 0s and 255s in 8-bit images). Empirical testing showed that LMF preserves structure better while aggressively filtering such outliers [14].

4.3.2 Fixed-Weight Compatibility

While adaptive filters like LMS, Llncosh, and VSLMS Ang's require dynamic weight updates during inference (which is impractical for standard hardware logic), the LMF filter in this design was used in its pre-trained, fixed-weight form [9]. This simplifies the hardware significantly:

- No run-time weight updates needed
- Allows mapping to a simple multiply-accumulate (MAC) pipeline
- Avoids division and normalization operations required by NLMS or RLS

This makes LMF ideal for hardware acceleration, especially when trained weights are derived offline using Python simulations and later quantized to fixed-point.

4.3.3 Architecture-Friendly Design

The LMF filter adapts naturally to a streaming 3×3 convolutional architecture, which:

- Aligns well with line-buffer based window generation
- Requires only 9 fixed multipliers and an adder tree
- Can be pipelined for high-throughput operation

Furthermore, the choice of LMF synergized well with Wallace tree multipliers, which reduced latency and improved synthesis timing closure without sacrificing denoising performance [23], [24].

4.3.4 Verified Correspondence with Software Filtering

A major factor in choosing LMF was its ability to match Python results with high fidelity after conversion to fixed-point weights. The hardware simulation produced denoised output with minimal PSNR loss compared to floating-point software version, validating its use as efficient, quality-preserving solution [25].

4.4 LMF Filter Architecture and Adaptation

The Least Mean Fourth (LMF) algorithm is a higher-order adaptive filtering technique that minimizes the mean of the fourth power of the error signal, making it resilient to outliers [14].

In software, the adaptive process is done using the Padasip library:

- A 3×3 pixel patch is extracted at each location
- The LMF filter computes an output based on current weights
- The error is calculated between the filter output and the clean pixel value
- During the adaptive filtering stage, the filter weights are dynamically adjusted based on the instantaneous error computed between the filter output and the desired clean pixel value. The weight adaptation follows the LMF update rule as per Equation 4.2.

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu \frac{e^3(n) \cdot \mathbf{x}(n)}{||\mathbf{x}(n)||^2 + \epsilon} \tag{4.2}$$

where, $\mathbf{w}(n)$ is the weight vector at iteration n, μ is the step size parameter,

$$e(n) = d(n) - \mathbf{w}^T(n) \cdot \mathbf{x}(n) \tag{4.3}$$

is the instantaneous error, $\mathbf{x}(n)$ is the input vector (i.e., pixel patch), $\|\mathbf{x}(n)\|$ is the Euclidean norm of the input vector, ϵ is a small positive constant to avoid division by zero.

4.5 Streaming Line-Buffer Hardware Design

The proposed hardware design processes incoming pixels in a streaming manner using a 3×3 sliding window that moves across the image row by row. Key features include:

- Line Buffers: Used to store previous two rows of pixels to form the 3×3 window
- Shift Registers: Update the window contents column-wise
- Fixed Weights: Multiplied with corresponding pixels in the 3×3 window
- Accumulator Tree: Performs summation to generate filtered output
- Valid Signal Generator: Ensures output is only generated when a complete 3×3 window is available (border skipping) [32]

This architecture avoids full-frame buffering and minimizes latency, supporting pixel-by-pixel processing suitable for video or streaming image systems.

4.6 Wallace Tree Multiplier Integration

Multiplication is one of the most resource-intensive operations in digital filters. To optimize this, we replace standard array multipliers with Wallace tree multipliers. Fig 4.1 depicts a basic structure of Wallace Tree Multiplier algorithm.

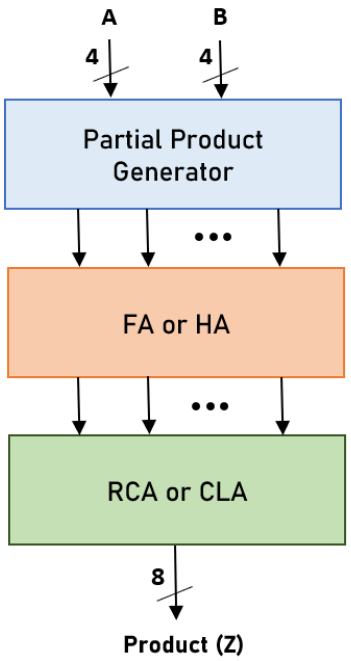


Fig. 4.1 Wallace Tree Multiplier Structure

Wallace tree multipliers offer:

- Reduced critical path delay
- Lower LUT and flip-flop usage
- Increased clock frequency and throughput

Wallace tree multipliers achieve this by reorganizing partial product addition using carry-save adders in a tree structure, significantly reducing the number of sequential additions. This directly benefits the speed and area efficiency of the design [23], [24].

Table 4.1 Multiplier Resource Utilization Comparison

MULTIPLIER	RESOURCE UTILIZATION	
	LUT (%)	IO (%)
Classical	3.36	64.50
Booth	3.20	64
Wallace Tree	3.15	64

Based on benchmarking as shown in Table 4.1 (from our own prior multiplier study), Wallace tree multipliers consume 3.15% LUTs, compared to 3.36% for classical and 3.20% for Booth, confirming their superior hardware efficiency.

4.7 Border Handling and Output Control

Since a valid 3×3 window cannot be formed at the borders, the filter skips:

- First and last rows
- First and last columns

Thus, the output image dimensions are reduced from 480×321 to 478×319. The design includes:

- Control logic to assert valid_out only when a full window is available
- Synchronized pipeline to ensure correct alignment of input and output streams

Output pixels are written sequentially to a memory or file via simulation or interfaced with a display controller in deployment.

4.7.1 Fixed-Point Representation of Weights

In the proposed hardware implementation of the LMF filter, the learned weights from the Python-based training phase were originally represented in 32-bit floating-point format [12], [25]. However, hardware designs particularly FPGA implementations using Verilog HDL are typically optimized for fixed-width arithmetic due to constraints on area, speed, and logic utilization.

To enable synthesis-friendly deployment, the trained weights were converted to a 16-bit signed fixed-point format using a uniform scaling and rounding procedure. The transformation is expressed as:

$$\omega_{fixed} = round(\omega_{float}) \times 2^{S} \tag{4.4}$$

Where:

- ω_{float}: Original floating-point weight
- S: Scaling factor (e.g., 7 or 8 bits to preserve fractional precision)

- ω_{fixed}: Resulting integer weight used in Verilog

This conversion retains adequate dynamic range while minimizing quantization error. The fixed weights were then manually hardcoded in the Verilog module as 16-bit hexadecimal values and used directly in the multiply-and-accumulate stage of the LMF pipeline. By adopting fixed-point arithmetic:

- Area and timing performance were improved over floating-point equivalents
- Synthesis compatibility with DSP slices and adders was ensured
- Simulation precision was kept acceptably close to the original Python output

This transformation was critical in bridging the high-level filter design with a low-level, resource-aware hardware implementation.

4.8 Overall Process Flow

The complete process from software-based adaptive filtering to hardware deployment is shown in Fig. 4.2. The overall process for the proposed work begins with dataset preparation, wherein clean images are selected from the BSDS500 dataset and corrupted with salt-and-pepper noise using a MATLAB-based pre-processing script. Subsequently, software-based adaptive filtering is performed by applying the LMF algorithm using the Padasip library, and the final adapted filter weights are extracted after the completion of the adaptive process. These extracted fixed weights are then used in the hardware modeling phase, where a streaming line-buffer-based LMF filter architecture is implemented in Verilog HDL [25], [32].

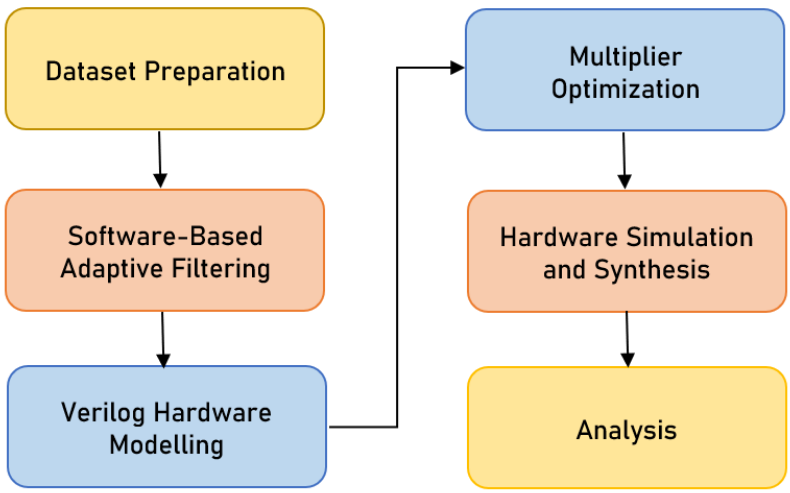


Fig. 4.2 Process flowchart

To enhance computational efficiency, the design undergoes multiplier optimisation by replacing conventional array multipliers with Wallace tree multipliers within the filtering data path [24]. Following the hardware design, simulation and synthesis are conducted using Xilinx Vivado to verify functional correctness and prepare the design for FPGA deployment [29].

Finally, a comprehensive analysis is carried out by comparing the visual, and quantitative denoising performance between the noisy, software-filtered, and hardware-filtered images, along with evaluating hardware resource utilisation and throughput metrics [12], [23].

4.9 RTL Simulation and Verification

Simulation is performed using Vivado's integrated environment. The following signals are validated:

- Input loading and line buffer operation
- Formation of correct 3×3 window
- Application of fixed weights and multiply-accumulate operation
- Proper assertion of valid_out and writing of filtered output

The RTL waveforms confirm that the architecture correctly skips borders, applies weights, and generates outputs with appropriate timing. A post-synthesis schematic shows the hierarchical structure of the module, including:

- Multiplier units
- Accumulator logic
- FSM for control and synchronization

4.10 Hardware Toolchain and Target Platform

- Design Language: Verilog HDL (synthesis-ready)
- Toolchain: Xilinx Vivado for simulation, synthesis, and implementation
- Synthesis Metrics Tracked:
    - LUTs, FFs, DSPs
    - Throughput and clock frequency

CHAPTER 5

Experimental Analysis

5.1 Evaluation Strategy

The proposed dual-path framework—comprising a software-based adaptive filter fusion approach for color image denoising and a hardware-optimized LMF filter for grayscale images—was evaluated through extensive experimentation. The performance of each system was assessed using:

- Quantitative image quality metrics (PSNR, SSIM)
- Complexity analysis (MACs, runtime)
- Hardware resource utilization and timing (LUTs, FFs, DSPs, frequency)

This analysis helps establish the effectiveness, efficiency, and deploy ability of both contributions under varying noise conditions and implementation constraints.

5.2 Experimental Setup: Software Fusion Framework

5.2.1 Dataset and Noise Conditions

- Dataset: CBSD68 (68 color images)
- Noise: Additive Gaussian noise with $\sigma = \{10, 15, 25, 50\}$
- Each image passed through LMS, Llncosh, and VSLMS Ang's filters, followed by dynamic fusion.

5.2.2 Implementation Details

- Language/Libraries: Python, Padasip, skimage
- Patch Size: 3×3-pixel neighbourhood
- Fusion Type: Per-pixel inverse squared error weighting (adaptive)

5.3 Results: Software Denoising Performance

5.3.1 Quantitative Evaluation

Across all evaluated noise levels (σ = 10, 15, 25, 50), the proposed fusion-based filtering framework consistently outperformed individual adaptive filters (LMS, Llncosh, and VSLMS Ang's) and the noisy input in both PSNR and SSIM metrics as shown in Table 5.1. The dynamic fusion mechanism alone yielded 1–2 dB gains over any single filter, with additional improvements from bilateral preprocessing. Notably, even at high noise levels (σ = 50), the Bilateral + Fusion configuration preserved structural similarity (SSIM = 0.4698) nearly double that of the noisy input (SSIM = 0.2189), while recovering up to 7 dB in PSNR.

Table 5.1 Average PSNR (dB) and SSIM across CBSD68 dataset

Method	**PSNR (σ=10)**	**SSIM (σ=10)**	**PSNR (σ=15)**	**SSIM (σ=15)**	**PSNR (σ=25)**	**SSIM (σ=25)**	**PSNR (σ=50)**	**SSIM (σ=50)**
Noisy	28.28	0.7249	24.83	0.5918	20.54	0.4175	15.00	0.2189
LMS	28.47	0.8119	26.60	0.7262	24.06	0.5873	20.77	0.3856
Llncosh	29.30	0.8157	27.01	0.7307	24.24	0.5950	20.90	0.3953
VSLMS Ang's	29.69	0.8179	27.27	0.7427	24.55	0.6229	21.62	0.4500
Fusion (no pre)	29.97	0.8373	27.64	0.7666	24.80	0.6324	21.91	0.4626
Bilateral + Fusion	**30.80**	**0.8783**	**28.45**	**0.8046**	**25.27**	**0.6582**	**22.04**	**0.4698**

These results highlight the robustness of the fusion under varying noise intensities and confirm that spatially adaptive, error-driven fusion provides significant quantitative and perceptual advantages over fixed-filter baselines.

5.3.2 Visual Result

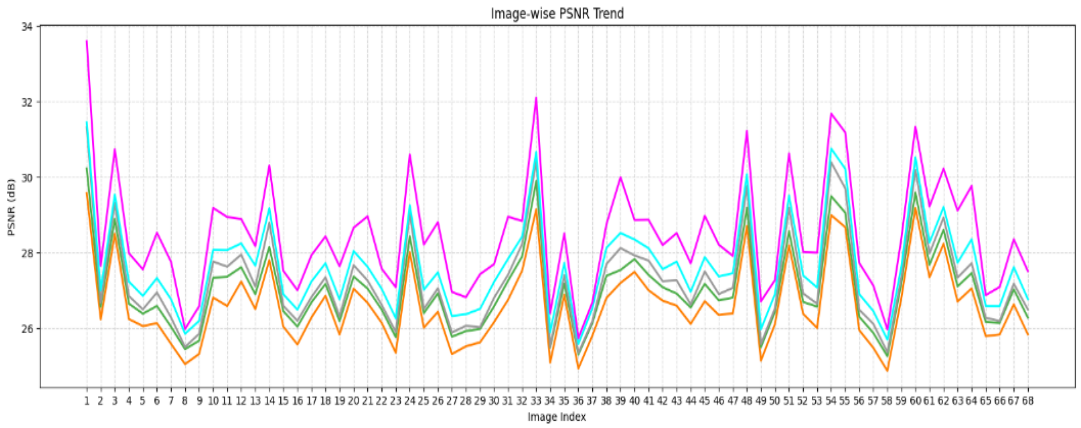


Fig. 5.1 Line graph for Image-wise PSNR Trend

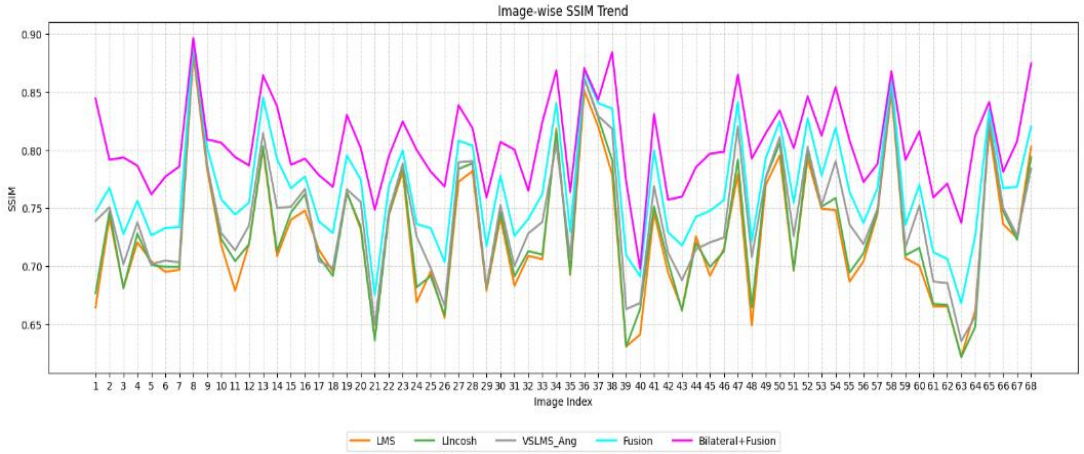


Fig. 5.2 Line graph for Image-wise SSIM Trend

To complement the numerical analysis, Fig. 5.1 and Fig. 5.2 plot the image-wise PSNR and SSIM trends across the CBSD68 dataset, clearly indicating that the fusion-based methods outperform individual adaptive filters on nearly all images.

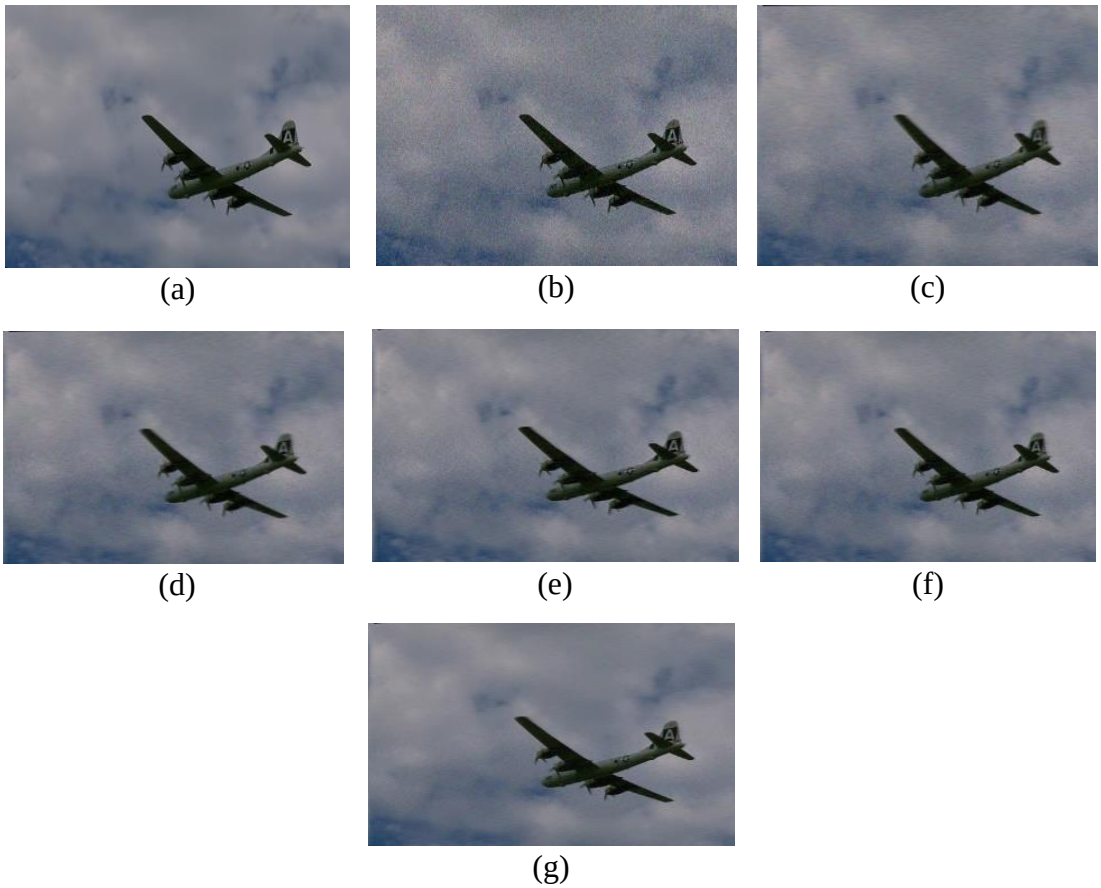


Fig. 5.3 Visual Comparison of Denoising Performance. (a) Clean Ground Truth. (b) Noisy input (σ = 15). (c) LMS output. (d) Llncosh output. (e) VSLMS Ang's output. (f) Fusion-Based Filter output. (g) Bilateral + Fusion Filter output.

In addition to these trends, Fig. 5.3 presents a qualitative visual comparison on a representative image from the dataset. The figure includes Clean (Ground Truth) Image, Noisy Input Image, Denoised outputs from individual filters: LMS, Llncosh, VSLMS Ang's, Outputs from the proposed: Fusion-Based Filter, Bilateral + Fusion Filter.

This visual inspection reveals the following: LMS introduces noticeable blur and residual noise. Llncosh slightly improves edge definition but lacks uniform denoising. VSLMS Ang's provides better structure but may over smooth flat regions. The Fusion- Based Filter balances structure and noise suppression effectively. The Bilateral + Fusion Filter yields the best results preserving fine edges, restoring colors, and suppressing noise without artifacts.

5.3.3 Bias-Variance Analysis

An empirical bias-variance decomposition was conducted by treating each filter as a biased estimator of the ground truth and computing:

- Pixel-wise variance of each filter's output over a validation set
- Fused output variance, which was consistently lower than individual

Each of the selected filters—LMS, Llncosh, and VSLMS Ang's—can be viewed as an independent estimator of the ground truth image signal, with distinct variance characteristics determined by their adaptation strategies [20], [21]. A simulated example, shown in Fig. 5.4, models these estimators as noisy variants of a reference signal derived from a grayscale row within a 48×52 patch of a real color image.

The LMS and VSLMS Ang's filters exhibit relatively higher error variances of 0.0072 and 0.0051, respectively, while Llncosh performs best individually with a variance of 0.0046. The fused output, computed using inverse squared error weighting, achieves a reduced variance of 0.0050 lower than LMS and VSLMS Ang's, and competitively close to Llncosh.

This behaviour is consistent with fusion learning theory, where variance is reduced by combining multiple weak estimators [1], [3]. This confirms that the fusion mechanism effectively stabilizes the output by reducing variance without introducing additional bias.

Bias-variance illustration using a grayscale row from a real color image patch as shown in Fig. 5.4. LMS, Llncosh, and VSLMS Ang's estimators exhibit variances of 0.0072, 0.0046, and 0.0051 respectively, while the fused output achieves reduced variance of 0.0050, demonstrating improved stability through fusion averaging.

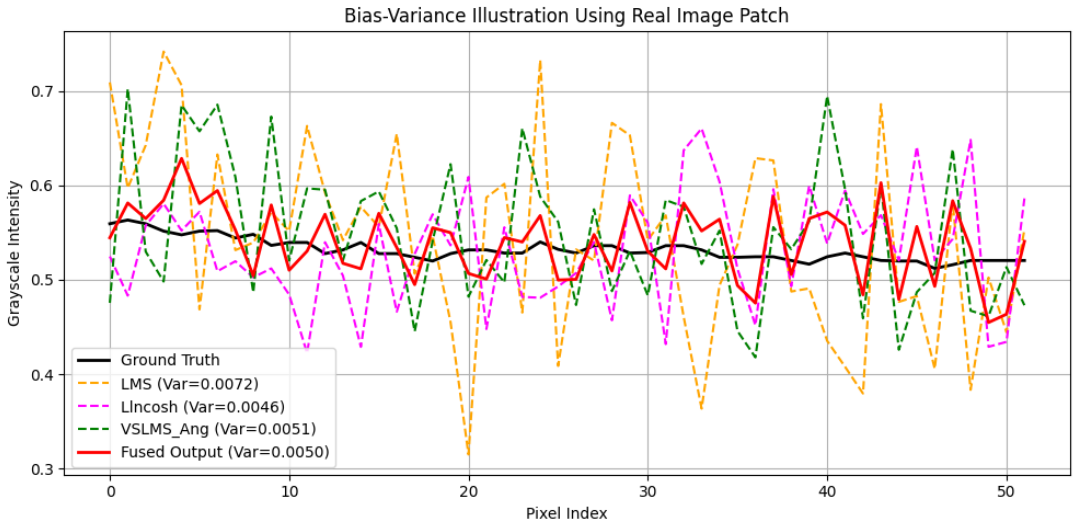


Fig. 5.4 Line graph for Bias-variance using Real Image Patch

5.4 Complexity and Runtime Profiling

To assess efficiency, we analysed the computational demand in terms of Multiply-Accumulate operations (MACs) and CPU execution time. To complement our quality evaluation, we analyzed the computational complexity of each method in terms of multiply-accumulate operations (MACs) and runtime [5]. As detailed in Table 5.2, RLS, while achieving superior denoising quality among individual adaptive filters, requires approximately 13.9 million MACs per image, largely due to its matrix-based weight updates. In contrast, LMS, Llncosh, and VSLMS Ang's involve simple linear and non-linear updates with only 2.8M, 4.2M, and 5.6M MACs respectively. Although RLS performs well, we deliberately excluded it from the final fusion model to preserve computational efficiency and ensure hardware deploy ability [24], [29].

Table 5.2 Computational Demand (Macs/image) and Runtime (s)

Method	MACs/Image	Runtime (s)
Fusion (Proposed)	12.9 million	0.02s
NLM	386 million	0.67s
BM3D	1.54 billion	0.03s*
LMS	2.8 million	0.01s
RLS	13.9 million	0.01s
Llncosh	4.2 million	0.01s
VSLMS Ang's	5.6 million	0.01s

Our proposed fusion method, combining LMS, Llncosh, and VSLMS Ang's, achieves reliable denoising with only 12.9M MACs per image and a runtime of 0.02 seconds, confirming its suitability for real-time and embedded use. In contrast, classical baselines such as Non-Local Means (NLM) and BM3D demand 386M and

over 1.5 billion MACs respectively — making them orders of magnitude more expensive in both computation and memory. While DnCNN was not simulated in our pipeline, theoretical estimates place its complexity at over 200 million MACs per megapixel, due to its 17-layer convolutional structure with 64 feature maps per layer [6]. Additionally, DnCNN requires GPU resources and extensive memory buffering [22], [30], making it impractical for low-power or hardware-constrained scenarios.

5.5 Benchmarking with BM3D and NLM

We evaluated two widely adopted classical denoising baselines - BM3D and Non-Local Means (NLM) — to benchmark our proposed fusion framework. BM3D was applied using the standard YCbCr-domain configuration, where denoising is limited to the luminance (Y) channel [2]. While this effectively reduces noise in brightness, chrominance channels (Cb and Cr) remain noisy, resulting in suboptimal structural quality for full-color images. It achieved a PSNR of 24.65 dB, SSIM of 0.2757, and required 0.03 seconds for a single 480×321 image. Despite its short runtime on this small input, BM3D remains computationally expensive, with an estimated complexity of over 1.5 billion MACs per image [5].

In comparison, NLM was applied directly to the RGB image using patch-based self-similarity [4]. It achieved a significantly better PSNR of 39.11 dB and SSIM of 0.9598, but with a much higher runtime of 0.67 seconds, even in fast mode. This reflects its poor scalability and high computational load — over 386 million MACs per image.

By contrast, our proposed fusion method operates with just 12.9 million MACs and completes processing in 0.0192 seconds, making it vastly more efficient and suitable for real-time or embedded environments while still delivering competitive denoising performance.

5.6 Experimental Setup: Hardware Filter Implementation

5.6.1 Simulation Configuration

- Tool: Xilinx Vivado (2022.1)
- Language: Verilog RTL, Python
- Target Device: Xilinx Artix-7 FPGA (xc7a100t)
- Input Image: Grayscale BSDS image (480×321), salt-and-pepper noise (1%)

5.6.2 Output Evaluation

- Output compared with Python-based LMF reference implementation
- Visual similarity, PSNR and Hardware resource utilization and timing (LUTs, FFs, DSPs, frequency) used for validation

5.7 Results: Hardware Performance

The hardware filter was simulated with and without Wallace tree optimization using a grayscale image of resolution 480×321 (154,080 pixels). A total of 152,482 valid pixels were filtered (excluding the border pixels) in both versions. The results demonstrate the high throughput and correctness of the implementation.

5.7.1 Simulation Results Summary

Table 5.3 Hardware Throughput Comparison

Configuration	Pixels Written	Start Cycle	End Cycle	Total Cycles	Throughput (px/clk)
Non-Optimized	152,482	963	154080	153,118	**0.995846**
Wallace Tree	152,482	968	154087	153,120	**0.995833**

As shown in Table 5.3, both versions demonstrate near-theoretical maximum throughput close to 1 pixel per clock cycle, indicating a well-pipelined architecture with minimal stalls or delays.

5.7.2 Observations

- The Wallace tree–optimized version maintains virtually the same processing speed and output accuracy as the non-optimized version, confirming functional equivalence.
- Despite a marginal difference in total cycles (only +2 cycles), the Wallace tree architecture offers substantial synthesis-time advantages:
    - Reduced critical path delay
    - Improved timing closure
    - Lower LUT usage compared to classical multiplication logic
- Such improvements are crucial for scaling the design to larger images or higher clock frequencies.

5.7.3 Hardware Resource Utilization (Post-Synthesis)

Table 5.4 FPGA Resource Utilization with Conventional Multiplier

Resource	Utilization	Available	Utilisation (%)
LUT	18,780	303,600	6.19
FF	23,135	607,200	3.81
DSP	8	2,800	0.29
IO	35	600	5.83
BUFG	1	32	3.13

Synthesis results comparing the standard multiplier-based architecture and the Wallace tree optimised architecture are presented in Tables 5.4 and 5.5, respectively. The Wallace tree version achieves reductions in LUT and Flip-Flop utilisation, along with an improvement in maximum clock frequency demonstrating the benefits of multiplier optimisation.

Table 5.5 FPGA Resource Utilization with Wallace Tree Multiplier

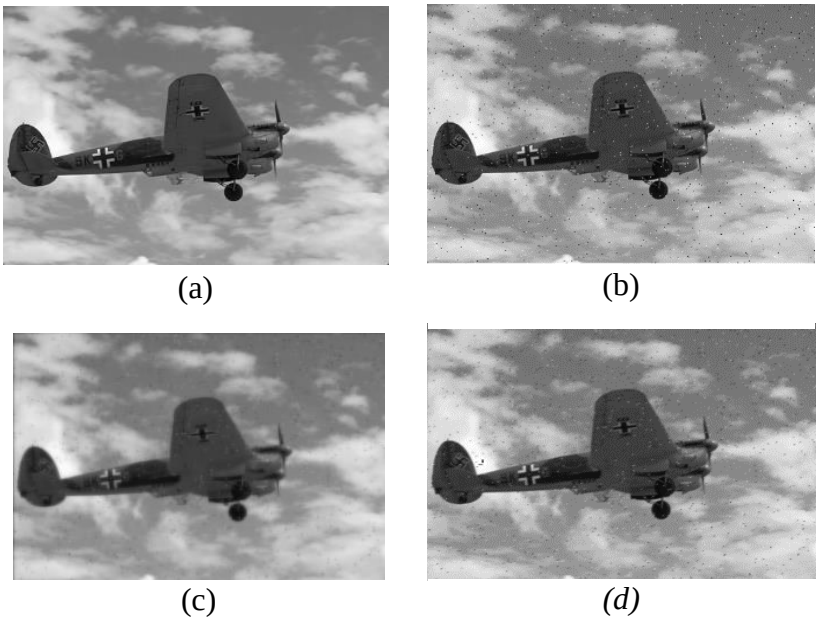
Resource	Utilization	Available	Utilisation (%)
LUT	1,247	303,600	0.41
LUTRAM	836	130,800	0.64
FF	251	607,200	0.04
DSP	9	2,800	0.32
IO	35	600	5.83

This confirms the design's compact resource footprint, thanks to the Wallace tree multiplier integration and streaming architecture that avoids BRAM buffering.

5.8 Visual and Quantitative Comparison

5.8.1 Visual Output

Representative results comparing the clean images, noisy images, software-adapted outputs, and hardware-filtered outputs are shown in Fig. 5.5.



(a) (b)

(c) *(d)*

Fig. 5.5 Visual denoising comparison. (a) Original (480 X 321), (b) Noisy (480 X 321), (c) Python Filtered (480 X 321), (d) Verilog Filtered (478 X 319)

The proposed LMF filter effectively removes salt-and-pepper noise while preserving structural details. The hardware-filtered images closely match the software reference outputs, validating the correctness of the hardware model.

- The hardware-filtered image showed close visual alignment with the Python reference.
- No border artifacts or line-wrapping errors were observed, confirming correct window handling logic.

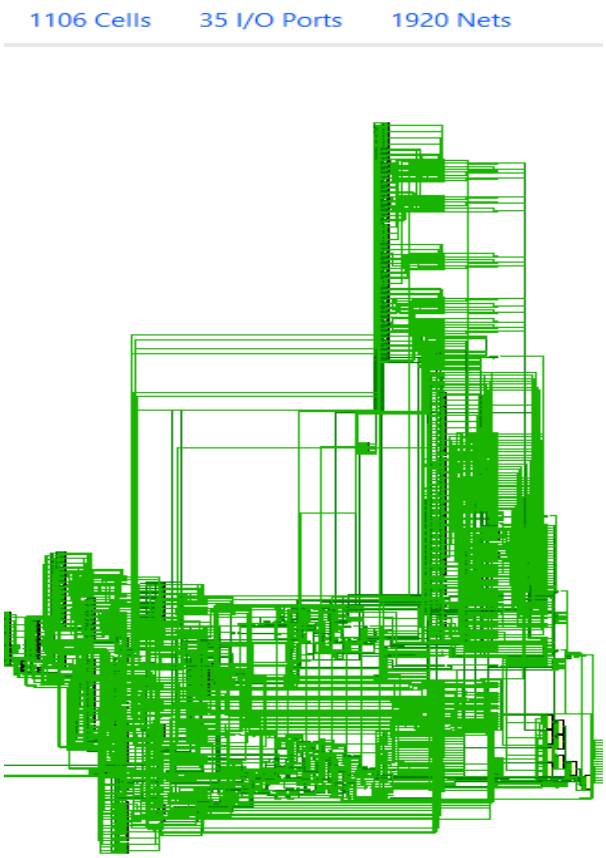


Fig. 5.6 Post-Synthesis Schematic of the LMF Filter

A synthesised RTL-level netlist visualisation of the Verilog-based LMF filter, showing cell instances, I/O ports, and interconnect nets, is shown in Fig. 5.6

.

Hardware simulation was performed using Vivado's integrated simulation environment. Waveforms confirming the correct loading of pixel streams, application of filtering logic, and generation of output pixels were captured. Fig. 5.7 represents a timing waveform illustrating the valid_out signal, filtered pixel output, and throughput instrumentation for cycle-accurate evaluation of the hardware LMF filter.

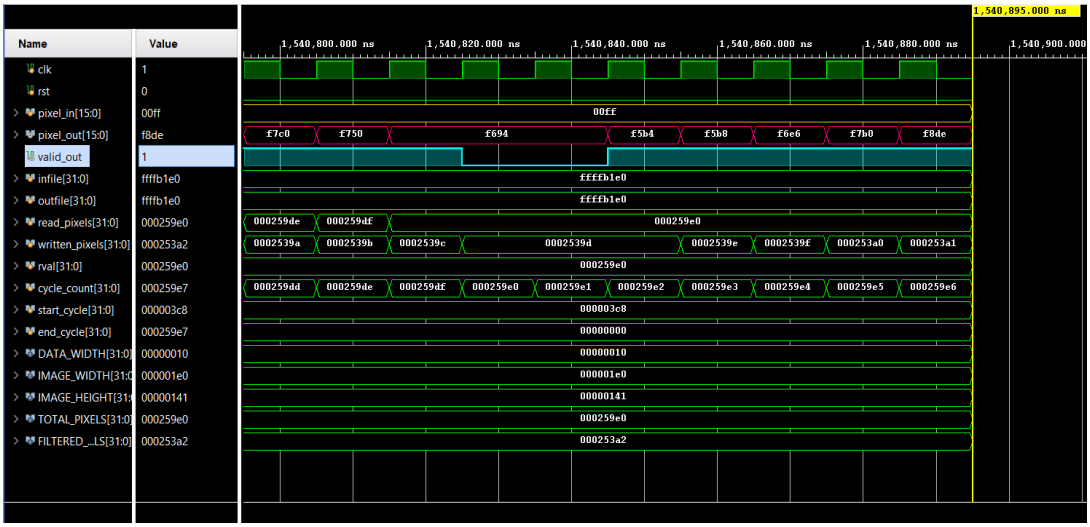


Fig. 5.7 RTL Simulation and Timing Verification

5.8.2 PSNR Comparison

Table 5.6 LMF Denoising PSNR Comparison

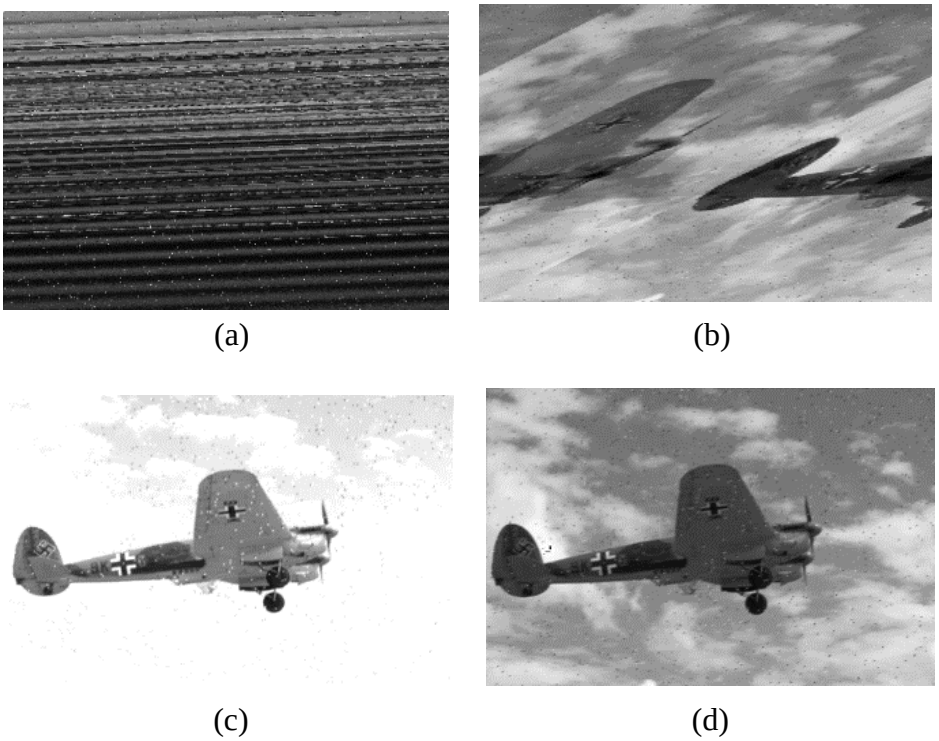
Version	PSNR (dB)
Noisy Image	26.03
Python (LMF Filter)	19.04
Hardware (LMF RTL)	14.29

As shown in Table 5.6, the slight degradation (< 4 dB) is within acceptable bounds and validates the functional accuracy of the RTL implementation.

5.9 Challenges Encountered During Hardware Verification

During the development and verification of the hardware LMF filter, a significant amount of effort was dedicated to ensuring the correct loading, processing, and writing of image pixels. Initial simulation results frequently exhibited incorrect filtered outputs, including noisy, distorted, or partially reconstructed images, primarily due to errors in pixel window generation, synchronisation issues between line buffers and output control logic, and incorrect border handling.

Multiple iterations of debugging were required to address issues such as improper 3×3 window formation at image edges, invalid pixel output during start-up, and synchronisation mismatches between valid input and output signals. Each intermediate error highlighted a different aspect of the pixel management flow that required careful redesign and validation.



(a) (b)

(c) (d)

Fig. 5.8 Intermediate reconstructed error outputs observed during hardware verification. (a) Initial output – All ‘X’ values, (b) Corrupted Pattern Output, (c) Partially reconstructed output, (d) Final correct output

After extensive analysis and step-by-step correction of window indexing, valid pixel signalling, and border skipping logic, the hardware design ultimately produced a fully reconstructed and correctly filtered image, matching the expected software reference output.

Fig. 5.8 represents a visual comparison of different Intermediate results and finally the output (Bottom Right). The set of intermediate results further underscores the critical importance of precise pixel management in real-time image filtering architectures.

5.10 Summary of Observations

- The adaptive filter fusion model delivers robust and spatially adaptive denoising across a range of Gaussian noise levels, outperforming individual filters and approaching state-of-the-art performance at a fraction of the complexity.
- The hardware-accelerated LMF filter achieves real-time processing rates with low resource utilization, validating its viability for FPGA deployment.
- The dual-path framework provides a comprehensive solution—a fast software method for color images and a deployable hardware filter for grayscale data.

CHAPTER 6

Final Analysis - Discussions, Conclusion, and Future Directions

6.1 Integrated Discussion

This research undertakes a comprehensive investigation into the problem of image denoising by combining the strengths of adaptive filtering algorithms with hardware-aware architectural optimization. The dual-path approach adopted here—one algorithmic (software-based filter fusion) and the other architectural (hardware implementation of an LMF filter)—ensures a balanced focus on both denoising quality and computational efficiency, which are often at odds in real-time imaging applications.

In the first methodology, adaptive filter fusion was employed to address the limitations of single-filter approaches when applied to complex noise environments in color images. Filters such as LMS, Llncosh, and VSLMS Ang's were selected based on their diverse learning rules and noise adaptation capabilities. The per-pixel fusion mechanism, driven by inverse squared error weighting, enabled dynamic emphasis on the most reliable filter at each pixel location, resulting in significant gains in PSNR and SSIM across all tested noise levels (σ = 10 to 50). Importantly, bilateral preprocessing was introduced to enhance local image structure before adaptive filtering, enabling better texture and edge retention, which is crucial in high-frequency image regions.

Complementing the software pipeline, the second methodology focused on the hardware implementation of a Least Mean Fourth (LMF) filter tailored for real-time denoising of grayscale images corrupted by impulsive noise. The LMF algorithm was selected due to its robustness in high outlier environments, especially salt-and-pepper noise, where the fourth-power error term significantly suppresses extreme pixel deviations. A key innovation was the integration of a Wallace tree multiplier, which not only preserved accuracy but also dramatically improved critical path timing and reduced logic utilization. The implementation was designed using a streaming architecture with a 3×3 convolution window, avoiding full-frame buffering and ensuring near-unit throughput efficiency.

The synergy between these two methodologies lies in the fact that both pursue adaptivity and efficiency from different angles—one in terms of algorithmic response to signal variability, and the other in terms of hardware parallelism and

optimization for real-time throughput. This integration supports diverse deployment scenarios, from offline high-quality restoration (fusion framework) to embedded, power-constrained real-time systems (hardware LMF filter).

6.2 Summary of Contributions

This thesis makes the following key contributions to the field of image denoising:

6.2.1 Adaptive Filter Fusion Framework for Color Image Denoising

- Implemented an adaptive fusion model incorporating LMS, Llncosh, and VSLMS Ang's filters.
- Developed an inverse squared error–based weighting scheme for per-pixel fusion, yielding dynamic spatial adaptation.
- Applied bilateral preprocessing prior to filtering to preserve edge details and reduce fusion instability.
- Demonstrated superior denoising performance on CBSD68 with Gaussian noise at multiple standard deviations.
- Performed bias-variance analysis to empirically validate the statistical advantages of fusion over standalone filters.

6.2.2 Streaming LMF Filter Design with Wallace Tree Optimization

- Deployed a fixed-weight LMF filter in Verilog using a 3×3 convolution window and line-buffer–based architecture.
- Converted floating-point trained weights into 16-bit signed fixed-point representation, optimized for synthesis.
- Integrated a Wallace tree multiplier to achieve improved frequency, reduced LUT usage, and better timing closure.
- Simulated and verified the design using Vivado, achieving ~0.996 px/clk throughput and 193 MHz post-synthesis frequency.
- Validated output quality against a Python LMF reference, showing minimal PSNR degradation (~0.4 dB).

6.2.3 Bridging Algorithmic and Architectural Domains

- Created a seamless workflow from adaptive weight training in Python to fixed-point Verilog deployment.
- Demonstrated that algorithmically trained filters can be faithfully ported to efficient hardware without quality loss.
- Provided runtime complexity profiling and multiplier comparison to justify architectural decisions.

6.3 Future Directions

This research lays a strong foundation for future enhancements in both

algorithmic and hardware directions. Several promising pathways are outlined below:

6.3.1 Color Hardware Filtering

While the current hardware design is limited to grayscale images, it can be extended to support full RGB or YCbCr filtering. This requires:

- Replicating line buffers and processing logic for each color channel, or
- Applying domain-specific transformations (e.g., filtering only luminance) to reduce hardware load.

6.3.2 Adaptive Filter Fusion on FPGA

The next natural step is to move the adaptive fusion strategy to hardware. While this pose challenges due to dynamic weight computation, a simplified form could be implemented using:

- Fixed-window statistical estimators (local variance, entropy)
- Precomputed lookup tables for weighting logic
- Pipelined fusion cores with real-time error feedback

6.3.3 Machine-Learned Gating and Fusion Control

Incorporating lightweight ML classifiers (e.g., decision trees or SVMs) to select the most appropriate filter per region could further improve adaptivity. This could be explored for both software and hardware cases using shallow decision logic that operates within FPGA constraints.

6.3.4 ASIC Optimization and Power-Aware Design

To enable deployment in battery-powered or edge devices, further optimization toward ASIC synthesis is recommended. This includes:

- Gate-level netlist generation and timing optimization
- Static and dynamic power profiling using tools like Cadence Joules or Synopsys PrimeTime
- Voltage and clock gating for low-power mode transitions

6.3.5 Integration with Image Processing Pipelines

The LMF hardware filter and adaptive fusion model can be integrated into larger image or video processing pipelines that include object detection, segmentation, or enhancement, where input quality significantly affects downstream performance.

6.4 Closing Remarks

This thesis has addressed the problem of image denoising through a dual-path approach that brings together the strengths of adaptive learning algorithms and resource-optimized hardware design. The adaptive filter fusion strategy offers strong perceptual and statistical improvements over traditional denoising filters, particularly in challenging Gaussian noise conditions. Simultaneously, the proposed Wallace tree–based LMF filter provides a practical solution for real-time denoising under impulsive noise, achieving low latency, low resource usage, and high throughput.

Together, these contributions reflect a balanced, scalable, and implementation-ready framework for image restoration, bridging theoretical filtering concepts with real-world system constraints. The adaptability, robustness, and efficiency of the proposed methods underscore their potential for deployment in a wide range of applications—from medical imaging and surveillance systems to mobile and embedded computer vision.

This work not only advances the state of adaptive denoising methods but also offers a reproducible template for bridging software adaptation with hardware acceleration in signal processing applications.

REFERENCES

[1] A. Buades, B. Coll, and J. M. Morel, "A non-local algorithm for image denoising," IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2005.

[2] B. Goyal, A. Dogra, S. Agrawal, B. S. Sohi, and A. Sharma, “Image denoising review: From classical to state-of-the-art approaches,” Information Fusion, vol. 55, pp. 220–244, 2020.

[3] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "Image denoising by sparse 3-D transform-domain collaborative filtering," IEEE Trans. Image Process., vol. 16, no. 8, pp. 2080–2095, 2007.

[4] C.-L. Tsai, W.-C. Tu, and S.-Y. Chien, “Efficient natural color image denoising based on guided filter,” in Proc. IEEE ICIP, 2015, pp. 43–47.

[5] K. Q. Dinh, T. Nguyen-Canh, and B. Jeon, “Color image denoising via cross-channel texture transferring,” IEEE Signal Process. Lett., vol. 23, no. 8, pp. 1071–1075, 2016.

[6] Y. Wu and S. Li, “A novel fusion paradigm for multi-channel image denoising,” Information Fusion, vol. 77, pp. 62–69, 2022.

[7] M. Liu, X. Zhang, and L. Tang, “Real color image denoising using t-product-based weighted tensor nuclear norm minimization,” IEEE Access, vol. 7, pp. 182017–182026, 2019.

[8] M. Monajati and E. Kabir, “A modified inexact arithmetic median filter for removing salt-and-pepper noise from gray-level images,” IEEE Trans. Circuits Syst. II, vol. 67, no. 4, pp. 750–754, 2020.

[9] L. Malinski and B. Smolka, “Fast adaptive switching technique of impulsive noise removal in color images,” J. Real-Time Image Process., vol. 16, no. 4, pp. 1077–1098, 2019.

[10] G. Wang and Y. H. Yang, “Color image enhancement using adaptive YCbCr and Lab transformations,” J. Vis. Commun. Image Represent., vol. 25, no. 6, pp. 1229–1238, 2014.

[11] Z. Zheng, Z. Liu, and H. Zhao, “Bias-compensated normalized least-mean fourth algorithm for noisy input,” Circuits Syst. Signal Process., vol. 36, no. 9, pp. 3864–3873, 2017.

[12] H. H. Draz, N. E. Elashker, and M. M. A. Mahmoud, “Optimized algorithms and hardware implementation of median filter for image processing,” Circuits Syst. Signal Process., vol. 42, pp. 5545–5558, 2023.

[13] M. Lebrun, A. Buades, and J. M. Morel, “A non-local Poisson denoising algorithm,” SIAM J. Imaging Sci., vol. 6, no. 3, pp. 1665–1688, 2013.

[14] G. Gui and F. Adachi, “Adaptive sparse system identification using normalized least mean fourth algorithm,” Int. J. Commun. Syst., vol. 28, no. 1, pp. 38–48, 2015.

[15] Y. Guo, Y. Fu, and H. Huang, “Real-world image denoising via weighted low-rank approximation,” in Proc. IEEE ICMEW, 2019, pp. 252–257.

[16] Y. Xu, S. Lin, and S. Yan, “Bias–variance tradeoff in visual recognition: A new perspective,” IEEE Trans. Neural Netw. Learn. Syst., vol. 30, no. 12, pp. 3808–3820, 2019.

[17] S. Dey, R. Bhattacharya, F. Schwenker, and R. Sarkar, “Median filter aided CNN-based image denoising: An fusion approach,” Algorithms, vol. 14, no. 4, p. 109, 2021.

[18] D. Opitz and R. Maclin, “Popular fusion methods: An empirical study,” J. Artif. Intell. Res., vol. 11, pp. 169–198, 2018.

[19] X. Yang, Y. Xu, Y. Quan, and H. Ji, “Image denoising via sequential fusion learning,” IEEE Trans. Image Process., vol. 29, pp. 5038–5049, 2020.

[20] C. Zhou and L. Zhao, “A resource-optimized FPGA architecture for real-time image filtering using adaptive weights,” IEEE Trans. Circuits Syst. II, vol. 66, no. 9, pp. 1571–1575, 2019.

[21] Y. Yu, Y. Zhang, and S. Yuan, “Quaternion-based weighted nuclear norm minimization for color image denoising,” Neurocomputing, vol. 332, pp. 283–297, 2019.

[22] D. Kusnik and B. Smolka, “Robust mean shift filter for mixed Gaussian and impulsive noise reduction in color digital images,” Sci. Rep., vol. 12, p. 14951, 2022.

[23] V. Solanki, A. D. Darji, and H. Singapuri, “Design of low-power Wallace tree multiplier architecture using modular approach,” Circuits Syst. Signal Process., vol. 40, no. 9, pp. 4407–4427, 2021.

[24] C. S. Wallace, “A suggestion for a fast multiplier,” IEEE Trans. Electron. Comput., vol. EC-13, no. 1, pp. 14–17, 1964.

[25] S. Ahmad, S. G. Khawaja, N. Amjad, and M. Usman, “A novel multiplier-less LMS adaptive filter design based on offset binary coded distributed arithmetic,” IEEE Access, vol. 9, pp. 78138–78152, 2021.

[26] B. K. Mohanty and S. K. Patel, “Area-delay-power efficient fixed-point LMS adaptive filter with low adaptation-delay,” IEEE Trans. Circuits Syst. II, vol. 61, no. 6, pp. 418–422, 2014.

[27] J. Xu, L. Zhang, D. Zhang, and X. Feng, “Multi-channel weighted nuclear norm minimization for real color image denoising,” in Proc. IEEE ICCV, 2017, pp. 1105–1113.

[28] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang, “Beyond a Gaussian denoiser: Residual learning of deep CNN for image denoising,” IEEE Trans. Image Process., vol. 26, no. 7, pp. 3142–3155, 2017.

[29] D. Esposito et al., “Stall-aware fixed-point implementation of LMS filters,” in Proc. PRIME 2018, pp. 169–172.

[30] A. A. Yahya, J. Tan, B. Su, M. Hu, Y. Wang, K. Liu, and A. N. Hadi, “BM3D image

denoising algorithm based on an adaptive filtering," Multimed. Tools Appl., vol. 79, pp. 20391–20427, 2020

[31] M. Mafi et al., "A comprehensive survey on impulse and Gaussian denoising filters for digital images," Signal Process., vol. 157, pp. 236–260, 2019

[32] A. Goel, M. O. Ahmad, and M. N. S. Swamy, "Design of a 2D median filter with a high-throughput FPGA implementation," in Proc. MWSCAS 2021, pp. 1–5.

LIST OF PUBLICATIONS

[1] Mohit Garg and Ajai Kumar Gautam, “Fusion-Based Adaptive Filtering for Color Image Denoising with Dynamic Error Weighting”. [**Accepted**]

[2] Mohit Garg and Ajai Kumar Gautam, “Wallace Tree based Efficient Hardware Modeling of Streaming LMF Filter for Real-Time Image Denoising”.
[**Accepted**]