Journal of VLSI circuits and systems
RESEARCH ARTICLE
Journal of VLSI Circuits and Systems, ISSN: 2582-1458 Vol. 7, No. 1, 2025 (pp. 176-185)
WWW.VLSIJOURNAL.COM

VLSI Implementation and Comparative Analysis of Anisotropic Diffusion Filters PMAD and FORADF for Real-Time Applications

Resmi R. Nair[1]*, R. Hema[2]
1Department of Computer Science and Engineering, Associate Professor, Saveetha Engineering College, Chennai, India.
[2]Department of Electronics and Communication Engineering, Associate Professor, Saveetha Engineering College, Chennai, India.

KEYWORDS:
VLSI
Anisotropic Diffusion
Hardware Complexity
Image Smoothing

ARTICLE HISTORY:
Received: 24.07.2025
Revised: 31.07.2025
Accepted: 08.08.2025

DOI:
https://doi.org/10.31838/jvcs/07.01.19

Abstract

In the era of miniaturization, designing an energy-efficient, high-quality image processing system for portable devices is a challenging task. Image processing and VLSI design are the two domains that can contribute toward achieving this goal. Images are often corrupted because of various types of noise during the different stages of image processing, from image acquisition to display. One of the widely occurring noises is the salt and pepper noise, which can be removed using median filtering. Median filtering preserves the edges, but it does not provide any smoothing effect. The visually appealing smoothing effect is very much essential in digital cameras nowadays. Anisotropic diffusion filters are well known for their smoothing effect with edge preservation. The foundational anisotropic diffusion filter, Perona & Malik anisotropic diffusion (PMAD), can remove Gaussian noise with edge preservation, but it failed under the perturbation of the salt and pepper noise. According to the literature, many anisotropic filters developed over decades for the removal of the salt and pepper noise from PMAD resulted in significantly higher hardware complexity compared to PMAD. The first-order robust anisotropic diffusion filter (FORADF) is well-suited for removing high-density salt and pepper noise with edge preservation and smoothing properties. The hardware complexity of FORADF is also much lower than PMAD. In this paper, the hardware implementation of PMAD and FORADF is discussed. RTL netlist, schematic view, and synthesis reports of both filters are generated using Xilinx Vivado 2023.1 and the Genus tool of Cadence. The comparison of the filters is conducted quantitatively and qualitatively. The comparative analysis undertaken in this paper shows that FORADF is well-suited for low-power, real-time applications.

Author's e-mail: resmi.gie@gmail.com; hemar@saveetha.ac.in

Author's Orcid id: 0000-0003-3733-0487; 0000-0003-3342-6369

How to cite this article: Nair R.R. and Hema, R., VLSI Implementation and Comparative Analysis of Anisotropic Diffusion Filters PMAD and FORADF for Real-Time Applications, National Journal of Antennas and Propagation, Vol. 7, No. 1, 2025 (pp. 176-185).

Introduction

With the increasing demand for compact and low complexity image processing systems in portable devices, the hardware implementation of image denoising algorithms has gained significant importance. In digital image processing, denoising of images has been a major concern. A lot of research has been done to develop filters that effectively reduce noise without sacrificing the important aspects of the image, such as textures and edges. Images may be corrupted because of the occurrence of various types of noise, such as Gaussian noise, Impulse noise, speckle noise, and so on. Gaussian noise can be removed using linear filters such as the mean filter and the Gaussian filter,[1] which are simple and fast, but they do not provide considerable edge preservation, resulting in extremely blurred images. In order to overcome these drawbacks, nonlinear filtering, such as the median filter,[2] was introduced. Nonlinear filters are used to suppress impulse noise and speckle noise by retaining the edge information intact. The salt and pepper noise is a kind of impulse noise that may occur during the process of image acquisition, image transfer, and image retrieval. The literature has proven the effectiveness of median filters in reducing the salt and pepper noise, and the nonlinear filters preserve

edges better than linear filters. However, the performance of the median filter is degraded significantly in the presence of high-density noise and across uniform sections in the image. Therefore, in real-time applications, a filter that can suppress noise as well as provide smoothing with edge preservation is desirable. Recent development trends to improve accuracy and robustness include fuzzy logic principles,[3] decision-based algorithms,[4] and adaptive median filters.[5] Despite these developments, the majority of these techniques may not be well-suited for implementation on real-time hardware because of their increased complexity or wide processing windows.[6-9]

Partial differential equation (PDE) techniques gained popularity because of their powerful edge-preserving and smoothing properties. Perona & Malik anisotropic diffusion (PMAD), originally developed by Perona and Malik, is a PDE-based method that iteratively optimizes the image by diffusing pixel intensities directionally.[10] PMAD is well recognized for its ability to perform selective smoothing; that is, diffusion will occur on the regions of uniform pixel values while the edges are preserved. Its capacity to preserve edges and fine details, and the ability to effectively eliminate noise, render it extremely applicable in medical imaging, surveillance, and remote sensing. The major drawback of anisotropic diffusion is the inefficiency of the algorithm under impulse noise perturbations. PMAD is the foundational anisotropic diffusion filter which serves as the basis for all other research developments in the class of anisotropic diffusion filters. According to the literature, many anisotropic diffusion filters, which are capable of suppressing the salt and pepper noise to some extent with larger computational complexity, have been suggested.[11-16] The computational intensity of these anisotropic diffusion filters is the major challenge toward real-time implementations.

A robust diffusion smoothing filter with low computational complexity has been developed.[17] The first-order robust anisotropic diffusion filter (FORADF) performs exceptionally well in Gaussian, impulse, and mixed noise scenarios. The inherent computational simplicity of the FORADF model is important for efficient VLSI implementations. Developing efficient VLSI architecture for real-time portable devices is a growing research area.[18-22]

In this paper, we propose VLSI architecture for the foundational PMAD and the low complexity FORADF. Further, a comparative study of the filters is performed based on the cell report, power report, and timing report. Section II provides an overview of PMAD and the VLSI implementation of PMAD. Section III provides an overview of FORADF and its VLSI implementation. A Comparative analysis of PMAD and FORADF is given in Section IV. Finally, the conclusion of the paper is given in Section V.

PMAD Filter

Perona & Malik (1990) introduced an anisotropic diffusion filter, PMAD, in which the diffusion coefficient is not a constant; instead, it is a function of directional gradients, $\nabla_D I$. The diffusion function is represented as $g(\nabla_D I)$, where D represents the north (N), south (S), east (E), and west (W) directions. The direction-dependent diffusion makes the filter anisotropic. This PMAD is efficient for the removal of Gaussian noise, but it fails in the presence of salt and pepper noise.

The numerical solution of the PMAD filter is expressed as

$$I_{i.j}^{n+1} = I_{i.j}^{n} + \lambda\left[g\left(\nabla_N I\right)\nabla_N I + g\left(\nabla_S I\right)\nabla_S I + g\left(\nabla_E I\right)\nabla_E I + g\left(\nabla_W I\right)\nabla_W I\right] \quad (1)$$

where $I_{i.j}^{n}$ represents the pixel value of the center pixel in a 3×3 window before filtering, λ is a constant, i,j

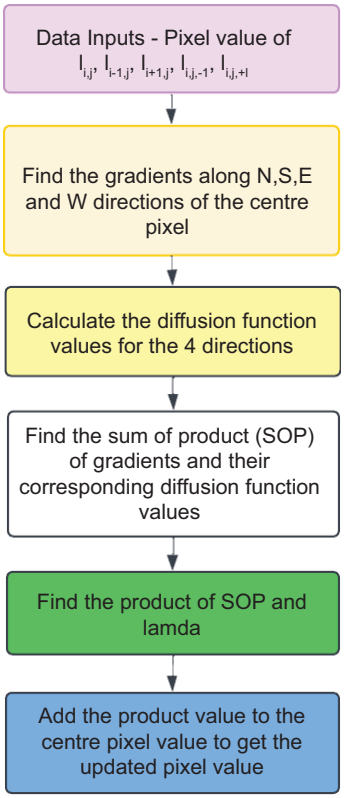

Fig. 1: A flowchart representing the computational steps involved in PMAD.

represent the spatial coordinates of the image pixel, and *n* represents the number of iterations.

$\nabla_N I$, $\nabla_S I$, $\nabla_E I$, and $\nabla_W I$ and represent the directional gradients along the north, south, east, and west directions of the center pixel $I^n_{i.j}$.

$$\nabla_N I = I^n_{i-1.j} - I^n_{i.j}, \cdots \nabla_S I = I^n_{i+1.j} - I^n_{i.j}$$
$$\nabla_E I = I^n_{i.j+1} - I^n_{i.j}, \cdots \nabla_W I = I^n_{i.j-1} - I^n_{i.j}$$

$I^{n+1}_{i.j}$ represents the updated pixel value of the center pixel after filtering.

The diffusion function, $\left(\nabla_D I\right) = e^{\left(-(\nabla_D I/K)^2\right)}$, where K is a constant.

Figure 1 shows a flow chart which represents the step-by-step computational process involved in PMAD. Figure 2A shows the register transfer level (RTL) netlist of the PMAD filter generated using the Xilinx Vivado

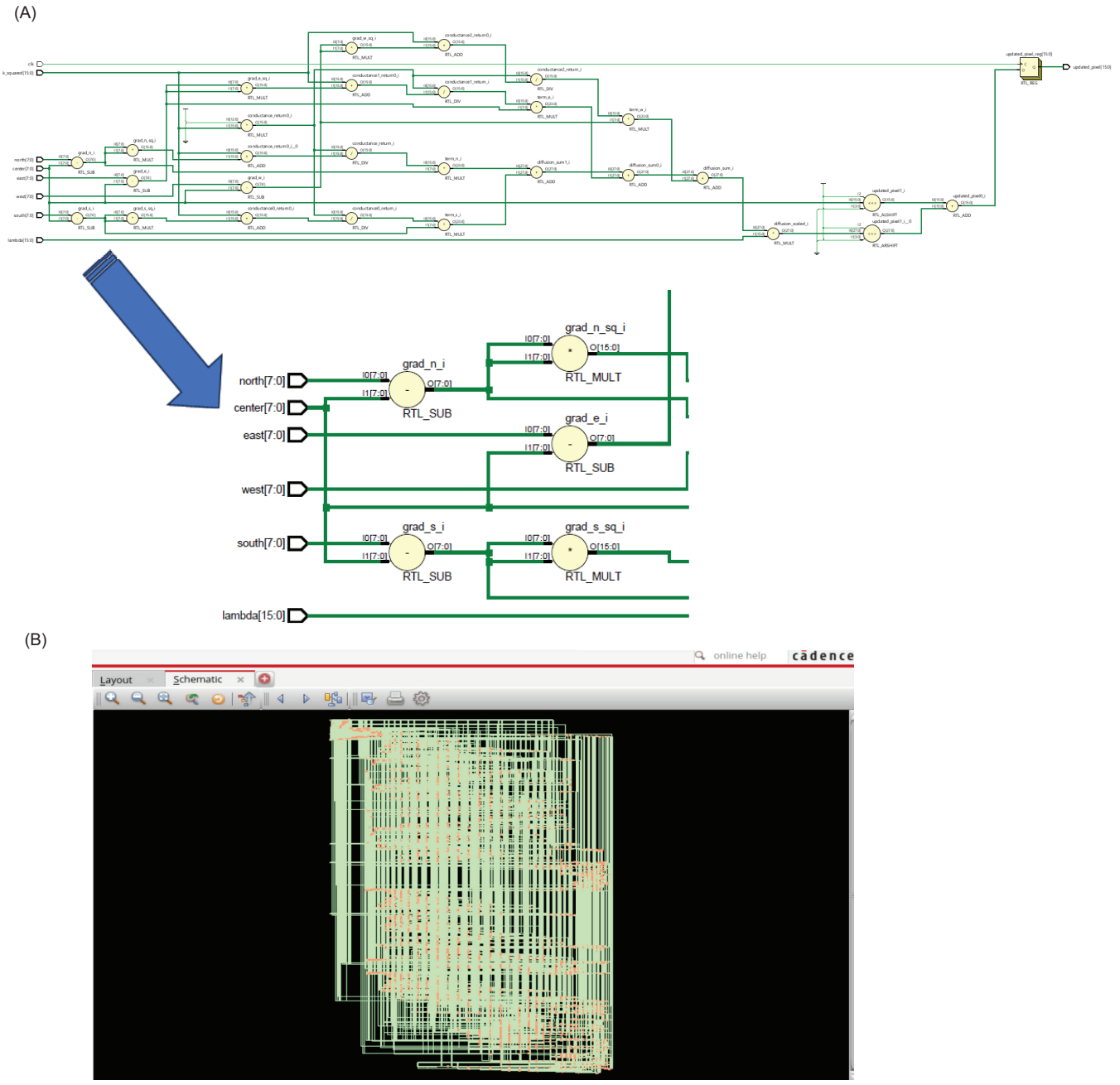

Fig. 2: (A) The RTL netlist of PMAD filter generated using the Xilinx Vivado 2023.1 tool consists of 44 cells, 89 I/O ports, and 587 nets. (B) The schematic of PMAD filter generated using the Genus tool in the The Cadence EDA tool consists of 4414 cells and 33851.595 μm² cell area.

2023.1 tool, which consists of 44 cells, 89 I/O ports, and 587 nets. The 44 cells include adders (RTL_ADD), subtractors (RTL_SUB), multipliers (RTL_MULT), dividers (RTL_DIV), arithmetic right shift (RTL_ARSHIFT), arithmetic left shift (RTL_ALSHIFT), and registers (RTL_REG). Figure 2B shows the schematic representation of the PMAD filter generated using the Genus tool in the Cadence EDA tool, in which the red color represents the functional cells and the green color lines represent the interconnections among various cells.

FORADF Anisotropic Diffusion Filter

Since the PMAD filter is not performing well under the perturbation of the salt and pepper noise, a new filter is modelled, and its performance is satisfactory even under the perturbation of high-density salt and pepper noise. The filter is designed with a reduced number of computational operations, such as addition and multiplication. Therefore, FORADF is a low-complexity filter. The median operation in the diffusion equation removes the salt and pepper noise inherently with image smoothing. The low-complexity FORADF provides image smoothing, edge preservation, and noise suppression.

The numerical solution for FORADF is

$$I_{i.j}^{n+1} = I_{i.j}^{n} + \lambda g\left\{Median\left(\nabla_N I,\ \nabla_S I,\ \nabla_E I,\ \nabla_W I\right)\right\}.$$
$$Median\left(\nabla_N I,\ \nabla_S I,\ \nabla_E I,\ \nabla_W I\right)$$

$$I_{i.j}^{n+1} = I_{i.j}^{n} + \lambda g\left\{Median\begin{bmatrix} I_{i-1.j}^{n} - I_{i.j}^{n}, \\ I_{i+1.j}^{n} - I_{i.j}^{n}, \\ I_{i.j+1}^{n} - I_{i.j}^{n}, \\ I_{i.j-1}^{n} - I_{i.j}^{n}, \end{bmatrix}\right\}.Median\begin{bmatrix} I_{i-1.j}^{n} - I_{i.j}^{n}, \\ I_{i+1.j}^{n} - I_{i.j}^{n}, \\ I_{i.j+1}^{n} - I_{i.j}^{n}, \\ I_{i.j-1}^{n} - I_{i.j}^{n}, \end{bmatrix} \quad (2)$$

where the diffusion function $g(.) = e^{-\left(Median\left(\nabla_N I,\ \nabla_S I,\ \nabla_E I,\ \nabla_W I\right)/2\right)}$, and λ is a constant. The exponential diffusion function is a function of the median of the directional gradients.

Figure 3 shows a flow chart which represents the step-by-step computational process involved in FORADF. Figure 4a shows the RTL netlist of the FORADF filter generated using the Xilinx Vivado 2023.1 tool, which consists of 25 cells, 74 I/O ports, and 252 nets. The 44 cells include adders (RTL_ADD), subtractors (RTL_SUB), multipliers (RTL_MULT), arithmetic right shift (RTL_ARSHIFT), asynchronous registers (RTL_REG_ASYNC), and a median (median4). Figure 4b shows the schematic

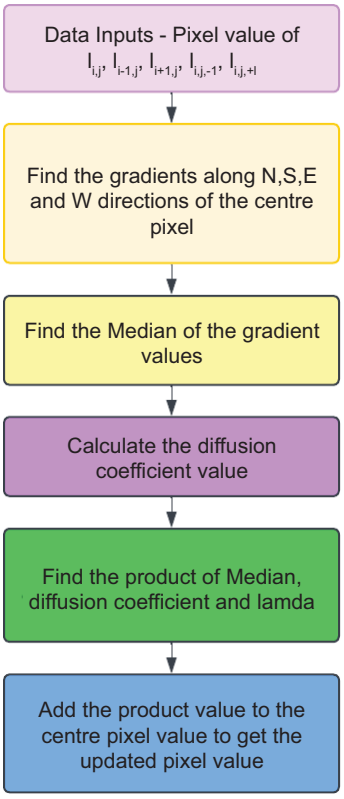

Fig. 3: A flowchart representing the computational steps involved in FORADF.

representation of the PMAD filter generated using the Genus tool in the Cadence EDA tool, in which the red color represents the functional cells and the green color lines represent the interconnections among various cells.

Results and Discussion

A comparative analysis of the PMAD and FORADF filters is discussed in this section. The mathematical models of the PMAD and FORADF filters are converted into Verilog HDL (Hardware Description Language), and the filters are synthesized using the Xilinx Vivado 2023.1 and using the Genus tool of the Cadence EDA platform. The RTL netlist view of the filters generated by the Xilinx Vivado 2023.1 synthesis tool is shown in Figure 2a and Figure 4a for PMAD and FORADF, respectively. A comparative analysis of the RTL netlist is performed in terms of the number of cells, I/O ports, and interconnections, and is shown in Figure 5. It is evident from the comparison that the FORADF is less complex than PMAD. The schematic of the filters generated by the Genus tool of the Cadence EDA platform is shown in Figure 2b and Figure 4b for PMAD and FORADF, respectively. It is clear from the schematic that the FORADF filter is much simpler

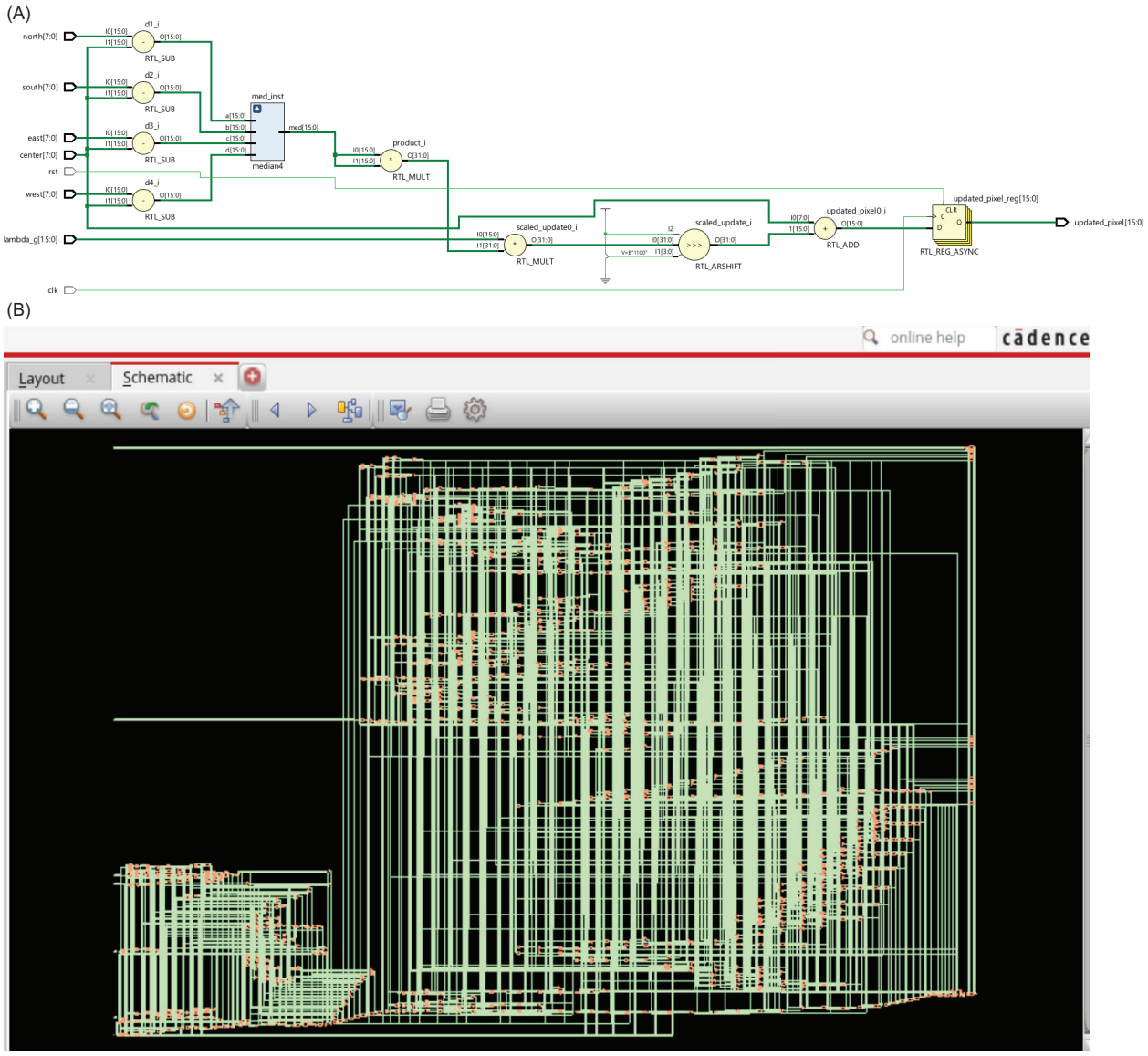

Fig. 4a: (A)The RTL netlist of the FORADF filter generated using the Xilinx Vivado 2023.1 tool consists of 25 cells, 74 I/O ports, and 252 nets. (B) Schematic of the FORADF filter generated using the Genus tool in the Cadence EDA tool consists of 944 cells and 7273.052 μm² cell area.

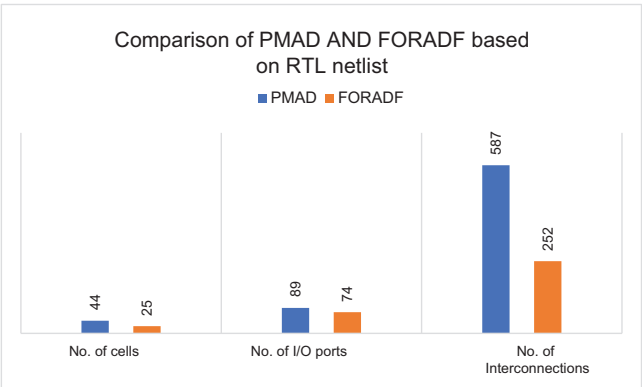

Fig. 5: Comparison of PMAD and FORADF based on the RTL netlist.

in terms of the number of cells and interconnections as compared to PMAD. The cell report, the power report, and the area reports are also generated using the Genus synthesis tool. Figure 6 and Figure 7 show the cell reports of PMAD and FORADF, respectively. The total number of cells and area required for PMAD are 4414 and 33851.595, respectively. Similarly, the total number of cells and area required for FORADF are 944 and 7273.052, respectively. The total number of cells and the area required for FORADF is much lower than PMAD. The power reports of PMAD and FORADF are shown in Figure 8 and Figure 9. The power report provides consumption details of leakage power, internal power, and switching power of the filters. The leakage

```
=============================================================
  Generated by:           Genus(TM) Synthesis Solution 21.14-s082_1
  Generated on:           Jun 23 2025  04:40:56 am
  Module:                 anisotropic_diffusion
  Technology library:     slow
  Operating conditions:   slow (balanced_tree)
  Wireload mode:          enclosed
  Area mode:              timing library
=============================================================

       Instance       Module  Cell Count  Cell Area  Net Area   Total Area   Wireload
-------------------------------------------------------------------------------------
anisotropic_diffusion              4414  33851.595     0.000    33851.595 <none> (D)
  (D) = wireload is default in technology library
```

Fig. 6: Cell report of PMAD.

```
=============================================================
  Generated by:           Genus(TM) Synthesis Solution 21.14-s082_1
  Generated on:           Jun 23 2025  03:06:15 am
  Module:                 pixel_update_squared
  Technology library:     slow
  Operating conditions:   slow (balanced_tree)
  Wireload mode:          enclosed
  Area mode:              timing library
=============================================================

      Instance       Module  Cell Count  Cell Area  Net Area   Total Area   Wireload
-------------------------------------------------------------------------------------
pixel_update_squared               944   7273.052     0.000     7273.052 <none> (D)
  (D) = wireload is default in technology library
```

Fig. 7: Cell report of FORADF.

```
 Instance: /anisotropic_diffusion
 Power Unit: W
 PDB Frames: /stim#0/frame#0
   ---------------------------------------------------------------------------
     Category         Leakage     Internal    Switching        Total    Row%
   ---------------------------------------------------------------------------
       memory     0.00000e+00  0.00000e+00  0.00000e+00  0.00000e+00   0.00%
     register     1.47433e-06  2.78020e-04  0.00000e+00  2.79494e-04   1.93%
        latch     0.00000e+00  0.00000e+00  0.00000e+00  0.00000e+00   0.00%
        logic     1.38265e-04  8.93169e-03  5.14369e-03  1.42136e-02  98.00%
         bbox     0.00000e+00  0.00000e+00  0.00000e+00  0.00000e+00   0.00%
        clock     0.00000e+00  0.00000e+00  1.06920e-05  1.06920e-05   0.07%
          pad     0.00000e+00  0.00000e+00  0.00000e+00  0.00000e+00   0.00%
           pm     0.00000e+00  0.00000e+00  0.00000e+00  0.00000e+00   0.00%
   ---------------------------------------------------------------------------
     Subtotal     1.39740e-04  9.20971e-03  5.15438e-03  1.45038e-02 100.00%
   Percentage           0.96%       63.50%       35.54%      100.00% 100.00%
   ---------------------------------------------------------------------------
```

Fig. 8: Power report of PMAD.

```
Instance: /pixel_update_squared
Power Unit: W
PDB Frames: /stim#0/frame#0
```

Category	Leakage	Internal	Switching	Total	Row%
memory	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00%
register	1.60723e-06	3.59910e-04	0.00000e+00	3.61517e-04	7.42%
latch	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00%
logic	2.69163e-05	2.85104e-03	1.62308e-03	4.50104e-03	92.36%
bbox	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00%
clock	0.00000e+00	0.00000e+00	1.06920e-05	1.06920e-05	0.22%
pad	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00%
pm	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00%
Subtotal	2.85235e-05	3.21095e-03	1.63377e-03	4.87324e-03	100.00%
Percentage	0.59%	65.89%	33.53%	100.00%	100.00%

Fig. 9: Power report of FORADF.

power, internal power, and switching power consumption of PMAD is 0.13974 mW, 9.20971 mW, and 5.15438 mW, respectively. Similarly, the leakage power, internal power, and switching power consumption of FORADF is 0.02852 mW, 3.21095 mW, and 1.63377 mW, respectively. The total power consumed by FORADF is 4.87 mW, which is significantly less than the total power consumed by PMAD, which is 14.50 mW. The timing reports of PMAD and FORADF are shown in Figure 10 and Figure 11, respectively. The total path delay for PMAD is 1398 ps, whereas that for FORADF is 1451 ps. Slack is positive for both PMAD and FORADF, and their slack values are 602 ps and 549 ps, respectively. Therefore, from the timing perspective, both designs met timing constraints with a positive slack.

Figure 12 shows the qualitative and quantitative performance of PMAD and FORADF at two different noise densities. The simulations are performed on MATLAB R2025a. The salt and pepper noise at 20% and 50% noise densities are considered for comparative analysis. Both filters are iterated only once, and $\lambda = 0.25$. The performance metrics peak signal-to-noise ratio (PSNR) and structural similarity index (SSIM) are calculated. The inherent median filtering property of FORADF made it more suitable for the suppression of the salt and pepper noise with image smoothing. The smoothing effect of anisotropic filters becomes more evident as the number of iterations increases. The superior performance of FORADF is evident in Figure 12.

A summary of the comparative analysis of the PMAD and FORADF implementations is shown in Table 1. The mathematical modelling of PMAD is given in Equation (1), and the mathematical modelling of FORADF is given in Equation (2). A deeper comparative analysis of these two equations reveals that the number of computational operations, mainly the number of additions and multiplications, required in PMAD is significantly higher than FORADF. Therefore, the total cell count, cell area, and power consumption are in turn reduced in the implementation of FORADF as compared to PMAD (Table 1). The comparative analysis of PMAD and FORADF in terms of the number of computational operations, the total cell count, cell area, total power consumption, total path delay, slack value, and the performance metrics (PSNR and SSIM) proves that FORADF is more suitable for VLSI implementation in comparison with PMAD. FORADF meets the two desirable requirements for VLSI implementation, reduced area and low power consumption, and hence, it can be considered as a suitable filter for real-time image processing applications.

Conclusion

The anisotropic diffusion filters PMAD and FORADF are discussed in detail. The RTL netlist for both filters has been generated using Xilinx Vivado 2023.1, and the results are compared. The schematic and synthesis reports for both the filters, PMAD and FORADF, are generated. A comparative analysis of the filters has been conducted based on the cell report, power report, and

```
╞=============================================================
  Generated by:           Genus(TM) Synthesis Solution 21.14-s082_1
  Generated on:           Jun 23 2025  04:40:56 am
  Module:                 anisotropic_diffusion
  Technology library:     slow
  Operating conditions:   slow (balanced_tree)
  Wireload mode:          enclosed
  Area mode:              timing library
=============================================================

             Pin                              Type       Fanout Load Slew Delay Arrival
                                                            (fF) (ps)  (ps)   (ps)
-------------------------------------------------------------------------------------------
(clock clk)                                  launch                                    0 R
updated_pixel_reg[0]/CK                                                  800    +0       0 R
updated_pixel_reg[0]/Q                       DFFQXL                1  0.0   26  +398     398 F
updated_pixel[0]                       <<<   interconnect                  26    +0     398 F
                                             out port                            +0     398 F
(constraints_top.sdc_line_5_15_1)            ext delay                         +1000    1398 F
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
(clock clk)                                  capture                                2000 R
-------------------------------------------------------------------------------------------
Cost Group   : 'clk' (path_group 'clk')
Timing slack :     602ps
Start-point  : updated_pixel_reg[0]/CK
End-point    : updated_pixel[0]
```

Fig. 10: Timing report of PMAD.

```
=============================================================
  Generated by:           Genus(TM) Synthesis Solution 21.14-s082_1
  Generated on:           Jun 23 2025  03:06:15 am
  Module:                 pixel_update_squared
  Technology library:     slow
  Operating conditions:   slow (balanced_tree)
  Wireload mode:          enclosed
  Area mode:              timing library
=============================================================

             Pin                              Type       Fanout Load Slew Delay Arrival
                                                            (fF) (ps)  (ps)   (ps)
-------------------------------------------------------------------------------------------
(clock clk)                                  launch                                    0 R
updated_pixel_reg[0]/CK                                                  800    +0       0 R
updated_pixel_reg[0]/Q                       DFFRHQX1              1  0.0   25  +451     451 F
updated_pixel[0]                       <<<   interconnect                  25    +0     451 F
                                             out port                            +0     451 F
(constraints_top.sdc_line_6_15_1)            ext delay                         +1000    1451 F
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
(clock clk)                                  capture                                2000 R
-------------------------------------------------------------------------------------------
Cost Group   : 'clk' (path_group 'clk')
Timing slack :     549ps
Start-point  : updated_pixel_reg[0]/CK
End-point    : updated_pixel[0]
```

Fig. 11: Timing report of FORADF.

Fig. 12: Performance of PMAD and FORADF on Lichtenstein test image for 20% and 50% salt and pepper noise densities.

Table 1: Summary of comparison between PMAD and FORADF implementations.

Parameters	Cell Count	Cell Area (μm^2)	Total Power (*mW*)	Total Path Delay (*ps*)	Slack Value (*ps*)	PSNR for 20% S & P noise	SSIM for 20% S & P noise
PMAD	4414	33851.595	14.5	1398	602	29.2824	0.9993
FORADF	944	7273.052	4.87	1451	549	33.0442	0.9999

timing report. Moreover, a performance comparison of the filters on a test image under the perturbation of salt and pepper noise at 20% and 50% noise densities is conducted. In terms of performance, power consumption, and complexity, FORADF is a more suitable choice for low-power, real-time applications. Furthermore, the same kind of comparative analysis can be conducted on any filter of our choice, and it will help implement them in real-time applications.

References

1. Gonzalez, R. C., & Woods, R. E. (2018). *Digital Image Processing* (4th ed.). Pearson.
2. Huang, T., Yang, G., & Tang, G. (1979). A fast two-dimensional median filtering algorithm. IEEE Transactions on Acoustics, Speech, and Signal Processing, 27(1), 13–18.
3. Hwang, H., & Haddad, R. A. (1995). Adaptive median filters: New algorithms and results. IEEE Transactions on Image Processing, 4(4), 499–502.
4. Srinivasan, K. S., & Ebenezer, D. (2007). A new fast and efficient decision-based algorithm for removal of high-density impulse noise. IEEE Signal Processing Letters, 14(3), 189–192.
5. Plataniotis, K. N., & Venetsanopoulos, A. N. (2000). *Color Image Processing and Applications*. Springer.
6. Poornimadarshini, S. (2024). Multi-layer security framework using RF fingerprinting and lightweight protocols in wireless networks. National Journal of RF Circuits and Wireless Systems, 1(1), 39–48.
7. Sadulla, S. (2024). State-of-the-art techniques in environmental monitoring and assessment. Innovative Reviews in Engineering and Science, 1(1), 25–29. https://doi.org/10.31838/INES/01.01.06
8. Rahim, R. (2024). Optimizing reconfigurable architectures for enhanced performance in computing. SCCTS Transactions on Reconfigurable Computing, 1(1), 11–15. https://doi.org/10.31838/RCC/01.01.03
9. Chia-Hui, C., Ching-Yu, S., Fen, S., & Ju, Y. (2025). Designing scalable IoT architectures for smart cities: Challenges and solutions. Journal of Wireless Sensor Networks and IoT, 2(1), 42–49.
10. Perona, P., & Malik, J. (1990). Scale-space and edge detection using anisotropic diffusion. IEEE Transactions on Pattern Analysis and Machine Intelligence, 12(7), 629–639.

11. Chen, X., Tang, C., & Yan, X. (2014). Switching degenerate diffusion PDE filter based on impulse-like probability for universal noise removal. AEU–International Journal of Electronics and Communications, 68(9), 851–857.
12. Khan, N. U., Arya, K. V., & Pattanaik, M. (2014). Edge preservation of impulse-noise filtered images by improved anisotropic diffusion. Multimedia Tools and Applications, 73(1), 573–597.
13. Tian, H., Cai, H., & Lai, J. (2014). A novel diffusion system for impulse noise removal based on a robust diffusion tensor. Neurocomputing, 133, 222–230.
14. Cho, S. I., Kang, S. J., Kim, H. S., & Kim, Y. H. (2014). Dictionary-based anisotropic diffusion for noise reduction. Pattern Recognition Letters, 46, 36–45.
15. Veerakumar, T., Esakkirajan, S., & Vennila, I. (2014). Edge-preservation adaptive anisotropic diffusion filter approach for the suppression of impulse noise in images. AEU–International Journal of Electronics and Communications, 68(5), 442–452.
16. Marami, B., Scherrer, B., Afacan, O., Erem, B., Warfield, S. K., & Gholipour, A. (2016). Motion-robust diffusion-weighted brain MRI reconstruction through slice-level registration-based motion tracking. IEEE Transactions on Medical Imaging, 35(10), 2258–2269.
17. Nair, R. R., David, E., & Rajagopal, S. (2019). A robust anisotropic diffusion filter with low arithmetic complexity for images. EURASIP Journal on Image and Video Processing, 2019, Article 48. https://doi.org/10.1186/s13640-019-0444-5
18. L. Wu and C. C. Jong. (2019). A High-Throughput VLSI Architecture for Real-Time Full-HD Gradient Guided Image Filter, IEEE Transactions on Circuits and Systems for Video Technology, 29(6), 1868-1877. https://doi.org/10.1109/TCSVT.2018.2852336.
19. Pal, C., Kotal, A., Samanta, A., Chakrabarti, A. and Ghosh, R., (2016). An efficient FPGA implementation of optimized anisotropic diffusion filtering of images. International Journal of Reconfigurable Computing, 2016(1), Article ID 3020473, pages 1-17, http://dx.doi.org/10.1155/2016/3020473
20. Aklak, A. F., Pugazhenthi, M. Y., & Alex, J. S. R. (2021). A study on VLSI implementation of image enhancement techniques. Concurrency and Computation: Practice and Experience, 34(10), Article e6734, 1–11. https://doi.org/10.1002/cpe. (If an article number is available, include it.)
21. AbdAlRahman, A., Al-Atabany, W. I., Soltan, A., et al. (2023). High-performance fractional anisotropic diffusion filter for portable applications. Journal of Real-Time Image Processing, 20, Article 98, 1–12. https://doi.org/10.1007/s11554-023-01339-y
22. Christudhas, C., & Fathima, A. (2024). VLSI implementation of a modified min–max median filter using an area and power competent tritonic sorter for image denoising. Scientific Reports, 14, 28628. https://doi.org/10.1038/s41598-024-80053-6