IIETA International Information and Engineering Technology Association
Advancing the World of Information and Engineering

Traitement du Signal
Vol. 41, No. 4, August, 2024, pp. 2135-2142
Journal homepage: http://iieta.org/journals/ts

An Optimized HW/SW Implementation of the Vector Median Rational Hybrid Filter for Real-Time Color Image Denoising

Ahmed Ben Atitallah

Department of Electrical Engineering, College of Engineering, Jouf University, Sakaka 72388, Saudi Arabia

Corresponding Author Email: abenatitallah@ju.edu.sa

https://doi.org/10.18280/ts.410441

Received: 23 November 2023
Revised: 26 February 2024
Accepted: 15 April 2024
Available online: 31 August 2024

Keywords:
nonlinear filter, VMRHF filter, real-time image denoising, HW/SW codesign, HLS flow, FPGA, low-latency implementation

ABSTRACT

The presence of noise in an image can significantly diminish its visual quality and adversely affect the accuracy of subsequent image processing tasks. Therefore, it is imperative to enhance image quality in real-time by eliminating disturbances introduced during the image acquisition or transmission process. This paper proposes an efficient and optimized implementation of the vector median rational hybrid filter (VMRHF) specifically tailored for real-time color image denoising. This filter is crafted to harness the capabilities of both the vector median filter and the rational operator, enabling effective noise reduction while maintaining the integrity of edges, image details, and chromaticity. However, the hybrid architecture in the VMRHF filter brings about an increase in computational complexity. To address this complexity, the filter is implemented in a Hardware/Software (HW/SW) codesign context, capitalizing on the strengths of both hardware and software components. The software component is created using the C/C++ programming language and operates on the ARM Cortex-A53 processor with a clock frequency of 1.2 GHz, while the high-level synthesis (HLS) flow is employed to develop the hardware portion, implemented as a coprocessor in the Zynq UltraScale+ XCZU9EG FPGA. Nevertheless, in the pursuit of crafting an optimized hardware architecture for VMRHF, specific directives like ARRAY_PARTITION and PIPELINE are progressively incorporated into the VMRHF C code using the Xilinx Vivado HLS tool. The interaction between the hardware and software parts is streamlined through the AXI-stream interface, facilitated by three Direct Memory Access (DMA) units for efficient data parallel transfer, thereby boosting data throughput. The VMRHF HLS design is evaluated on the embedded ZCU102 kit. The experimental outcomes illustrate that our design is capable of restoring a 256×256 color image within 19 ms, reflecting a substantial 94% decrease in execution time compared to the software design. This notable improvement is achieved while upholding consistent image quality, as indicated by both objective measures such as peak signal-to-noise ratio (PSNR) and subjective assessments. These results hold true across various levels of "salt and pepper" impulsive noise. Besides, our design exhibits a power consumption of merely 4.46 watts.

1. INTRODUCTION

Image denoising is a technique used in image processing to remove noise from images [1, 2]. Noise, in the context of images, refers to unwanted variations in color that might be created by a variety of causes, including transmission interference, sensor limitations, and environmental conditions. By removing these undesirable artifacts and keeping crucial features, image denoising aims to improve an image's visual quality.

Nonlinear filters [3, 4] are a type of image filter that operates on an image's pixel values in a way that is not a linear combination of the input pixel values. Unlike linear filters, which compute weighted averages, nonlinear filters use more complex functions to determine the output pixel value. Nonlinear filters are particularly useful for tasks such as image enhancement, edge preservation, and noise reduction.

The VMRHF [5] is one of the most popular nonlinear filters. This filter is used in various image processing applications, including color image denoising, enhancement, and restoration [6-8]. However, removing "salt and pepper" impulsive noise from a color image using this filter is challenging due to its high algorithmic complexity and time-consuming nature.

In the existing literature, various approaches have been suggested to minimize the computational complexity of the VMRHF filter with the objective of enabling real-time image restoration. In fact, Khriji et al. [9] propose a hardware architecture for the VMRHF that is implemented on a Field Programmable Gate Arrays (FPGA) circuit. In this architecture, some approximations are introduced to implement the rational function and the division operation. In this work, the VMRHF architecture allows for the processing of 40 million RGB vectors per second at 40 MHz. Boudabous et al. [10] suggest a HW/SW implementation of the VMRHF filter. Indeed, a fixed-point is used for the hardware

implementation. In contrast, the NIOS II softcore processor is employed to execute the software part. So, the proposed design allows for processing an image of size 176×144 pixels in 38 ms at 60 MHZ. Bernacchia et al. [11] present a programmable hardware architecture for the MRHF's. The floating-point operations are implemented using some approximations. The performance of this architecture depends on the image size and the frame rate. Khriji and Gabbouj [12] introduces a method for multichannel image processing through an adaptive approach, which is suggested to be more suitable and straightforward for implementing the VMRHF filter in software.

In this study, we introduce a novel and enhanced HW/SW codesign implementation for the VMRHF filter. Nevertheless, the advantage of HW/SW codesign lies in its ability to optimize the overall system performance by seamlessly integrating both hardware and software components. This approach allows for efficient distribution of tasks between dedicated hardware and flexible software, leading to enhanced speed, power efficiency, and overall system functionality [13-15]. However, the HLS flow is employed for the hardware implementation. In fact, the HLS offers a promising approach for accelerating hardware design and improving productivity by using a high-level programming language, like C, C++ or System instead of the traditional hardware description languages (HDLs) [16, 17]. The purpose of the HLS is to automate the process of converting high-level code into synthesizable hardware. But the challenge with the HLS is that the designers need to iterate through several experiments with different pragmas and directives to achieve the desired performance. Conversely, the software component is run on the ARM Cortex-A53 hardcore processor. Thus, employing the codesign approach in this study enables a harmonized utilization of hardware acceleration via HLS for computationally intensive tasks, and software flexibility through the Cortex-A53 processor for dynamic and adaptable functions. This leads to enhancements in performance, resource utilization, and power efficiency.

Hence, the goal of this research is to develop an efficient VMRHF filter design capable of real-time color image denoising while optimizing power efficiency and resource utilization. Therefore, the proposed design will be implemented and assessed in terms of execution time, FPGA cost, power consumption and image quality using the embedded Zynq UltraScale+ MPSoC ZCU102 kit.

The subsequent sections of the paper are organized as follows: Section 2 introduces the VMRHF filter. Section 3 describes the VMRHF hardware architecture, which is designed through the HLS flow. Section 4 illustrates the VMRHF HW/SW design and the evaluation of its performance on the ZCU102 kit. Section 5 concludes the paper.

2. VMRHF FILTER

Figure 1 presents the block diagram of the VMRHF filter. This filter contains two stages. In the first stage three vector median filters (VMFs) [18, 19] are applied to the three 3×3 filter window. In the second stage, the outputs of the three VMFs filter (Φ_1, Φ_2 and Φ_3) are provided to the vector rational function (VRF) to compute the filtered pixel $y(n)$ by the following formula:

$$y(n) = \Phi_2(n) \tag{1}$$

$$+\frac{\sum_{i=3}^{3}\beta_i\Phi_i(n)}{h + k.\|\Phi_1(n) - \Phi_3(n)\|}$$

where, the vector $\beta = [\beta_1, \beta_2, \beta_3]^T$ characterizes the constant vector coefficient of the input sub-functions which satisfy to the following condition $\sum_{i=1}^{3}\beta_i = 0$. In our case, we selected $\beta = [1, -2, 1]^T$. The h and k parameters are positive constants. The parameter k is utilized to control the quantity of the nonlinear effect [20]. Φ_2 is a center weighted median sub-filter whereas sub-filters Φ_1, and Φ_3 are selected for an adequate compromise between noise reduction and chromaticity and edge preservation can be reached. In this case both these sub-filters (Φ_1, and Φ_3) operate as bidirectional vector median filters. The former applies its action on a plus-shaped mask, while the latter employs a cross-shaped one. $\|.\|_2$ is the L_2 vector norm and so that:

$$= \sqrt{(\Phi_{11} - \Phi_{31})^2 + (\Phi_{12} - \Phi_{32})^2 + (\Phi_{13} - \Phi_{33})^2} \tag{2}$$

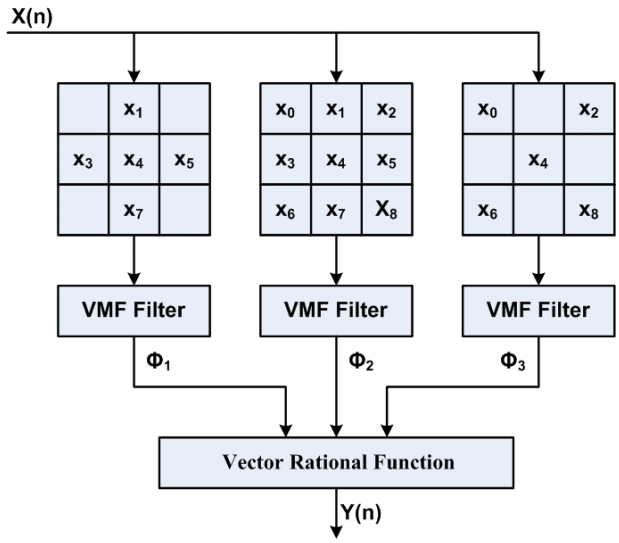

Figure 1. VMRHF block diagram

Nevertheless, the VMRHF filter utilizes both the VMF filter and rational functions. The rational function structure is particularly advantageous for filtering due to its universal approximation capability and effective extrapolation abilities. It yields the best approximations for certain specific functions. On the other hand, VMFs demonstrate strong performance when dealing with noise that adheres to a long-tailed distribution, such as impulsive noise. Additionally, VMFs easily identify and eliminate outliers in image data. Consequently, the VMRHF output is a result of a vector rational operation applied to the output of three sub-filters (Φ1, Φ2, and Φ3). This filter showcases desirable characteristics, including the preservation of edges and details, as well as accurate chromaticity estimation.

3. VMRHF HARDWARE ARCHITECTURE

HLS provides numerous benefits concerning productivity, portability, abstraction, and design simplicity. In fact, HLS enables designers to utilize higher-level programming languages like C, C++, or SystemC, allowing them to articulate their designs in a more abstract manner. This

abstraction reduces the intricacy of the design process, resulting in enhanced productivity. Additionally, code composed in high-level languages tends to be more readable and maintainable. The hardware generated by HLS can leverage these characteristics, making it more comprehensible and modifiable as needed. Moreover, describing the design with a high-level language in HLS enhances potential portability across various hardware platforms. The tool handles the translation of the high-level description into hardware specific implementations. HLS also expedites development cycles by enabling designers to swiftly prototype and experiment with diverse algorithms and architectures at a high level before committing to a final design. This accelerates the exploration of design spaces. Furthermore, HLS tools can automatically perform optimizations during the synthesis process. These optimizations encompass aspects such as clock speed, resource utilization, and power consumption, which may be challenging and time-consuming to achieve manually. In summary, HLS emerges as a potent methodology, especially in applications where rapid development, design exploration, algorithmic optimizations, and high-level abstractions are paramount.

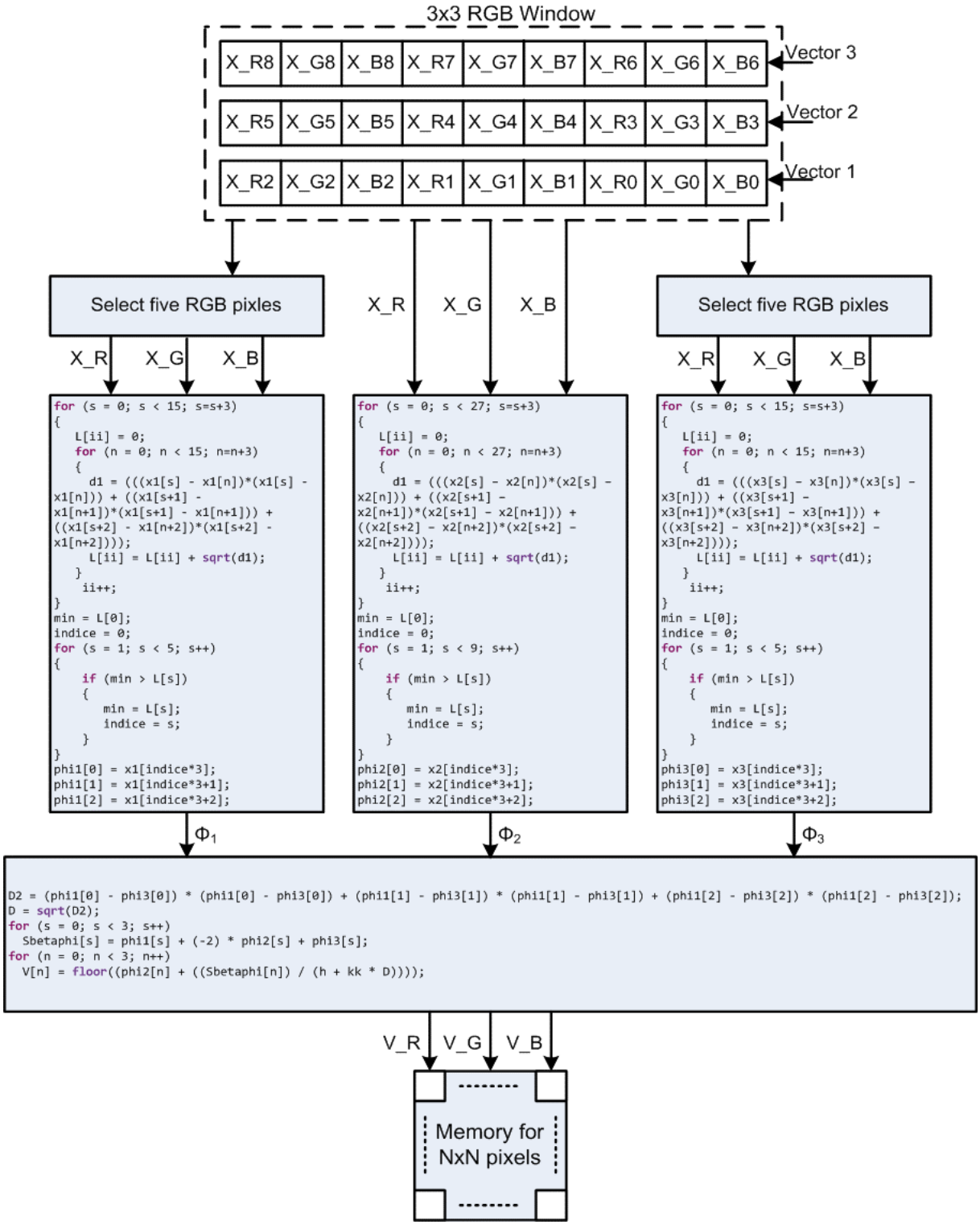

Figure 2. VMRHF HLS design

In this context, numerous HLS tools have been developed in this context, including the Xilinx Vivado HLS tool. This tool is part of the Xilinx Vivado Design Suite, which is a comprehensive development environment for FPGA and SoC (System on Chip) designs. Vivado HLS allows designers to work at a higher level of abstraction, describing hardware functionality in terms of algorithms and behaviors using C, C++, or SystemC and automatically generating the Register Transfer Level (RTL) code in Verilog or VHDL. Moreover, it includes a range of optimizations to improve the performance, area utilization, and power consumption of the generated hardware. It performs optimizations such as pipelining, loop unrolling, and resource sharing. Further, it can exploit dataflow and task parallelism in the high-level code, automatically generating parallel hardware structures that can lead to improved performance [21, 22].

In our work, Xilinx Vivado HLS v2018.1 is used. Figure 2 presents the HLS architecture generated for the VMRHF filter. In fact, a 3×3 filter window is provided in the input. Then, the selector is used to select the adequate RGB pixels for each of the three VMF filters. Afterwards, the three VMF filters are processed in parallel to compute Φ_1, Φ_2, and Φ_3, which are provided in parallel in the input of the VRF to calculate the filtered RGB pixel. The filtered color image is then reconstructed by storing each RGB pixel in the internal memory. The advantages of this architecture are that it supports floating-point operations and uses parallel and pipeline techniques to increase performance.

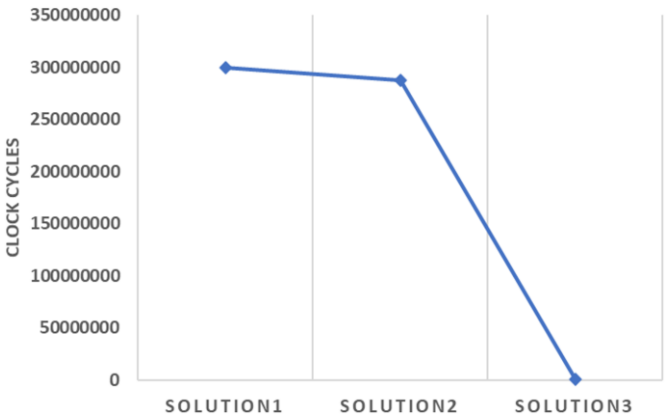

Figure 3. Evaluation of clock cycles number

Nevertheless, in order to enhance the performance of the VMRHF HLS design, multiple directives are progressively incorporated into the VMRHF C code, leading to the generation of diverse hardware solutions. Thereby, solution #1 is generated without adding any directives. This solution needs 300019722 clock cycles to process a 256×256 color image, as depicted in Figure 3, and occupies in the Zynq UltraScale+ XCZU9EG FPGA 5% of look-up tables (LUTs), 1% of flip-flops (FFs), 14% of BRAM blocks, and 2% of DSP slices, as presented in Table 1 and Figure 4. From these results, it is clear that this solution needs a huge amount of time to process a 256×256 color image. For this, to reduce this time, the ARRAY_PARTITION directive is applied to the VMRHF C code. This directive is used to guide the tool on how to partition arrays and can be beneficial for optimizing parallel access to the arrays and improving the performance of the hardware accelerator. Thus, solution #2 is created by implementing the ARRAY_PARTITION directive on the three filter windows within the VMRHF C code. This enables parallel access to the RGB pixels by the VMF filter, resulting in a reduction in the time required to compute the RGB filtered pixel. The synthesis results show that this solution can decrease the clock cycle numbers by 4% compared to solution #1, as Figure 3 illustrates, with only a 1% increase in the number of FFs, as shown in Figure 4. But the data throughput still remains low in solution #2 for the VMRHF design. Therefore, we suggest using the PIPELINE directive to increase the data throughput of the design by breaking down the computation into stages and allowing multiple stages to operate concurrently. This is crucial for achieving high clock frequencies and improving performance in FPGA designs. In our scenario, we employ the PIPELINE directive to guide Vivado HLS in the pipelining of a loop with an initiation interval of 2. This enables the simultaneous processing of all RGB pixels within a single line of an image. Consequently, while the PIPELINE directive can decrease the time required to process the entire image, it also leads to an escalation in hardware resource usage due to the multiplication of processing blocks. Accordingly, solution #3 is generated. This solution permits a reduction of the clock cycle numbers by 99% compared to solution #2, but with an increase in the FPGA resources as represented in Figure 4. Nevertheless, our goal is to process a color image in real-time with the VMRHF filter. For that reason, we selected solution #3 for HW/SW codesign implementation of the VMRHF filter. This solution allows for the processing of 13 Mpixels/s.

Table 1. Hardware resources used by each solution on the XCZU9EG FPGA

	Solution1	Solution2	Solution3
LUT	13708	14036	65730
FF	8151	10542	51725
BRAM_18K	124	118	198
DSP48E	50	50	701

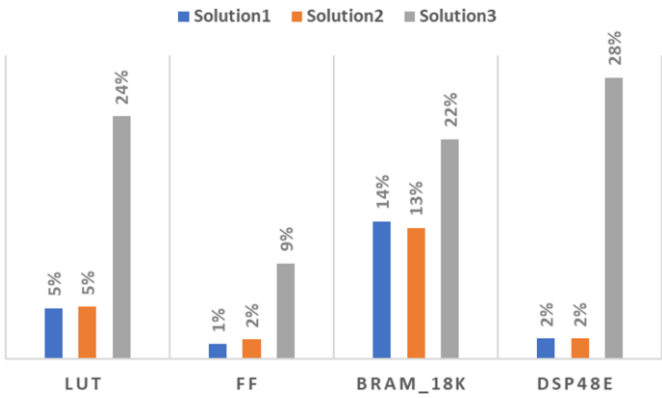

Figure 4. Percentage comparison of the hardware resources hardware on the XCZU9EG FPGA

4. VMRHF IMPLEMENTATION

4.1 HW/SW codesign

HW/SW codesign involves the simultaneous development and optimization of both hardware and software components within a system, aiming to achieve specific performance, power, and hardware cost objectives. The architecture of the VMRHF HW/SW design is depicted in Figure 5. In this design, the software component is built upon the ARM Cortex-A53

processor, operating at a clock frequency of 1.2 GHz. On the other hand, the hardware component incorporates the VMRHF coprocessor. To facilitate communication between the software and hardware components, the AXI-Stream interface is employed, utilizing three Direct Memory Access (DMA) controllers to enhance data throughput [23]. Indeed, these DMAs are used to manage the transfer of the RGB pixels from or to the DDR memory of the processing system. Specifically, DMA1 operates in read/write mode, while DMA2 and DMA3 are exclusively operated in read mode. Configuration of these DMAs is accomplished through the AXI-Lite interface.

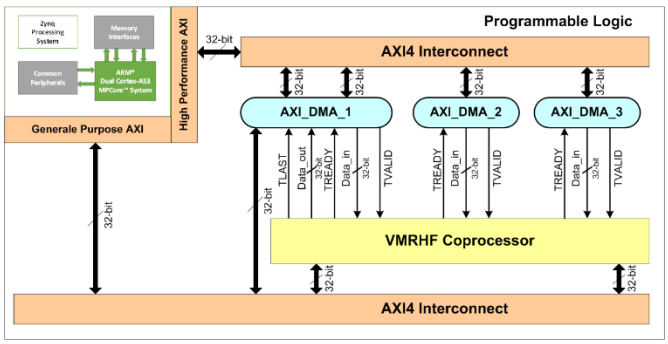

Figure 5. VMRHF HW/SW design

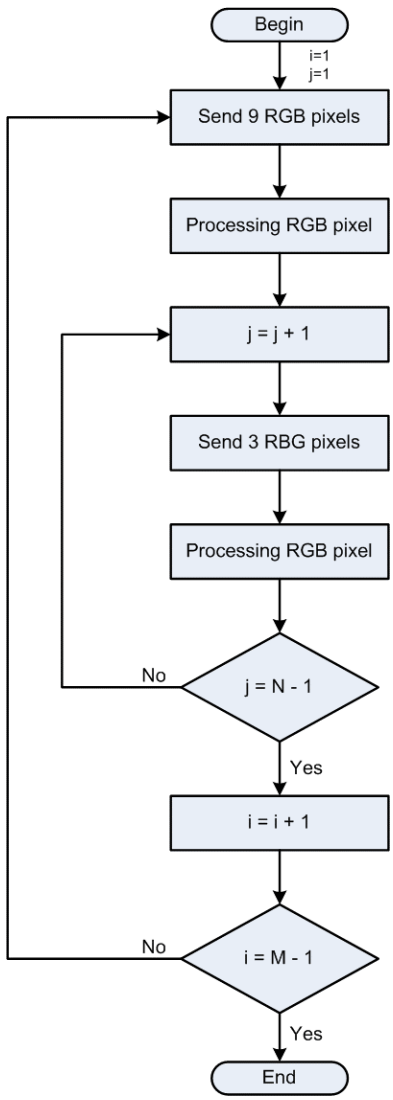

Figure 6. Data transfer management

Figure 6 presents the principles of data transfer management. In fact, in the beginning, the three DMAs are used to transfer the three first lines of the image to the VMRHF coprocessor. When the nine RGB pixels of the filter window are ready, the VMRHF coprocessor starts to process these pixels in order to determine the filtered pixel. Then, only three RGB pixels are selected to reconstruct the next filter window by adding these pixels to the six RGB pixels of the previous filter window. This technique is used until the image's RGB pixels are processed by the VMRHF coprocessor, which allows for optimization of the data transfer time.

In our architecture, the internal memory is used to store each processed RGB pixel once it has been concatenated into a 24-bit representation. Afterwards, the DMA1 is employed to move every pixel to the processing system's DDR memory so that the filtered image may be restored.

The implementation results of the VMRHF HW/SW design on the Zynq UltraScale+ XCZU9EG FPGA by the Xilinx Vivado v2018.1 tool show that our design needs 25.02% of LUTs, 11.65% of FFs, 6.91% of BRAM blocks, and 27.82% of DSP slices, as depicted in Table 2. Thereby, our design keeps enough space in the FPGA to add other hardware blocks.

Table 2. FPGA cost of the VMRHF HW/SW design

Resource	Utilization	Available	Utilization in %
LUTs	68565	274080	25.02%
FFs	63872	548160	11.65%
BRAM	63	912	6.91%
DSP	701	2520	27.82%

4.2 Performance evaluation

The evaluation of the VMRHF HW/SW design's performance is conducted on the Zynq UltraScale+ MPSoC ZCU102 kit, which is represented in Figure 7. The Zynq UltraScale+ XCZU9EG FPGA is the kit's central component. The ARM Cortex-A53 processor is located within this FPGA. However, the function XTime_GetTime() uses this processor's timer to determine the execution time. Moreover, different standard color images are employed to measure the performance of the VMRHF design. The resolution of these color images is 256×256 pixels. Additionally, the PSNR [24], which is obtained by Eq. (3), is used to determine the objective measurement of the image.

$$PSNR = 10\log\left(\frac{255^2}{MSE}\right) \quad (3)$$

where, the Mean Square Error (MSE) is given by Eq. (4).

Figure 7. Zynq UltraScale+ MPSoC ZCU102 kit

$$MSE_k = \frac{1}{NM}\sum_{i=1}^{N}\sum_{j=1}^{M}(o_k(i,j) - x_k(i,j))^2 \qquad (4)$$

where, $o_k(i,j)$ is a pixel in the original image, $x_k(i,j)$ is a pixel in the restored image, k characterizes the color channel and N and M characterize the image size.

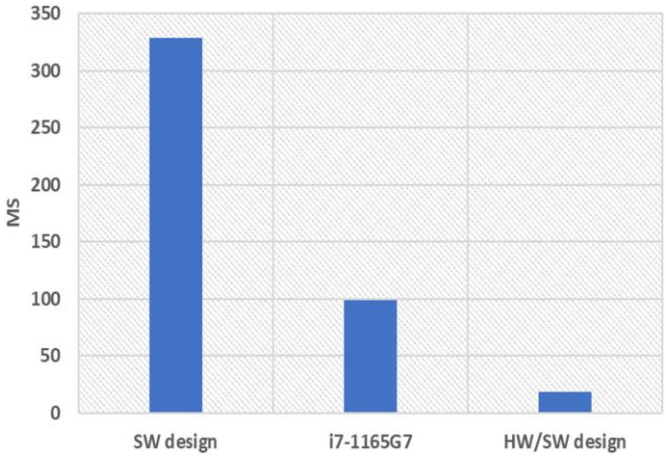

Figure 8. Comparison of the execution time

Figure 8 presents the comparison of the execution times of the VMRHF filter under the Cortex-A53 processor (SW design), the Intel CoreTM i7-1165G7@2.80GHz processor, and the HW/SW design. It is clear from this figure that the VMRHF HW/SW design allows a decrease of 94% and 82% in the execution time relative to the SW design and i7-1165G7@2.80GHz processor, respectively. The results are provided for comparable objective and subjective image quality in both the software (SW) and HW/SW codesign implementations, as shown in Figure 9. Additionally, as depicted in Figure 10, our VMRHF HW/SW design exhibits a power consumption of merely 4.46 watts.

Table 3 shows the performance comparison of our VMRHF designs with the works presented in researches [9, 10]. Indeed, our VMRHF HW/SW architecture accomplishes a 53% reduction in execution time while exhibiting a 67% increase in power consumption in comparison to the approach presented in study [10]. It is noteworthy that study [10] relies on a NIOS II processor operating at a clock frequency of 60 MHz, interfacing with the VMRHF coprocessor using fixed-point precision. Conversely, our design incorporates a Cortex-A53 processor operating at 1.2 GHz, interfacing with the VMRHF coprocessor utilizing floating-point precision. As a result, in comparison to study [10], our solution achieves a notable decrease in execution time, maintains image quality and details, albeit with a heightened power consumption. This power increase is justified by a substantial 95% rise in clock frequency relative to the configuration in study [10]. We will compare now our VMRHF HLS design with study [9]. The Low-Level Synthesis (LLS) is employed in study [9] to implement the VMRHF filter on hardware. In this architecture, approximations are introduced to implement the rational function and the division operation. Consequently, this architecture is implemented through fixed-point operations, resulting in a reduction in the quality of the filtered image when compared to our HLS design, which employs floating-point operations. So, despite the fact that our HLS design needs more time to filter a color image than study [9], it can process the image in real-time with conserving the quality and details of the image.

Figure 9. Comparing image quality between the SW and HW/SW implementations of the VMRHF filter (a) original image, (b) corrupted image by 5% of impulsive noise, filtered image by (c) SW design and (d) HW/SW design

Table 3. Performance comparison

Reference	Image Size	Precision	Technology	Execution Time	Throughput	Power Consumption
[9]	--	Fixed-point	LLS	--	40 Mpixels/s	--
[10]	176×144	Fixed-point	NIOS II+VMRHF coprocessor	38 ms	0.6 Mpixels/s	1.47 watts
Proposed	256×256	Floating-point	HLS	5 ms	13 Mpixels/s	--
	256×256	Floating-point	Cortex-A53+VMRHF coprocessor	18 ms	3 Mpixels/s	4.46 watts

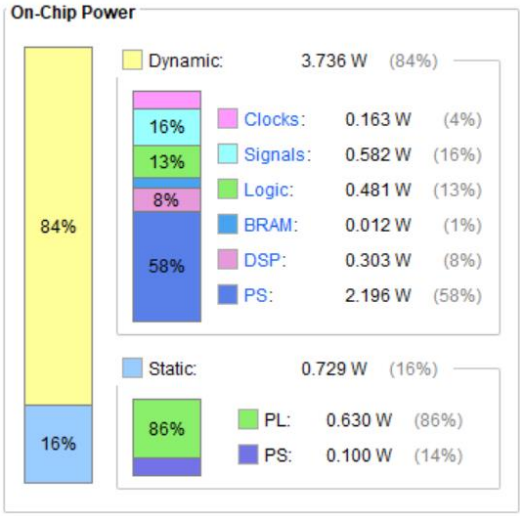

Figure 10. On-chip power

5. CONCLUSIONS

In this research, we introduce an effective HW/SW design for the implementation and evaluation of the VMRHF filter. The HLS is utilized to generate an optimized hardware architecture with floating-point precision, employing the VMRHF C code. This process incorporates diverse directives such as ARRAY_PARTITION and PIPELINE. In fact, the ARRAY_PARTITION feature facilitates parallel access to RGB pixels in the filter window, leading to a reduced computation time for RGB filtered pixels. Additionally, the PIPELINE functionality enables simultaneous processing of all RGB pixels in a single line of an image, thereby decreasing the time needed to process the entire image. This architecture enables the processing of a 256×256 color image in 5 ms. Subsequently, the architecture of the VMRHF is incorporated as a coprocessor alongside the ARM Cortex-A53 processor within the Zynq UltraScale+ XCZU9EG FPGA. To improve data throughput, three DMAs facilitate the connection between the hardware and software components. Experimental results obtained on the embedded ZCU102 kit demonstrate that the VMRHF HW/SW design achieves a remarkable 94% reduction in execution time relative to the SW design while maintaining equivalent objective and subjective image quality. These outcomes affirm the effectiveness of the developed VMRHF filter design.

The suggested VMRHF filter system finds application in video processing, particularly in scenarios where denoising is employed to improve the visual quality of videos. Additionally, the VMRHF filter proves valuable in security and surveillance applications, where denoising is crucial for refining the clarity of surveillance images, facilitating easier identification of objects or individuals.

As future work, we propose incorporating the VMRHF filter into video standards such as High Efficiency Video Coding (HEVC) and Versatile Video Coding (VVC). This integration aims to elevate video quality, leading to enhanced the compression and streaming quality.

REFERENCES

[1] Fan, L., Zhang, F., Fan, H., Zhang, C. (2019). Brief review of image denoising techniques. Visual Computing for Industry, Biomedicine, and Art, 2(1): 7. https://doi.org/10.1186/s42492-019-0016-7

[2] Girdher, A., Goyal, B., Dogra, A., Dhindsa, A., Agrawal, S. (2019). Image denoising: Issues and challenges. In Proceedings of International Conference on Advancements in Computing & Management (ICACM-2019), pp. 404-410. https://doi.org/10.2139/ssrn.3446627

[3] Peltonen, S., Kuosmanen, P. (2001). Robustness of nonlinear filters for image processing. Journal of Electronic Imaging, 10(3): 744-756. https://doi.org/10.1117/1.1380388

[4] Lorenzo-Ginori, J.V., Plataniotis, K.N., Venetsanopoulos, A.N. (2002). Nonlinear filtering for phase image denoising. IEE Proceedings-Vision, Image and Signal Processing, 149(5): 290-296. https://doi.org/10.1049/ip-vis:20020626

[5] Khriji, L., Gabbouj, M. (1999). Vector median-rational hybrid filters for multichannel image processing. IEEE Signal Processing Letters, 6(7): 186-190. https://doi.org/10.1109/97.769365

[6] Dang, D., Luo, W. (2008). Color image noise removal algorithm utilizing hybrid vector filtering. AEU-International Journal of Electronics and Communications, 62(1): 63-67. https://doi.org/10.1016/j.aeue.2007.02.001

[7] Khriji, L., Gabbouj, M. (2002). Generalised class of nonlinear-type hybrid filters. Electronics Letters, 38(25): 1650-1651. https://doi.org/10.1049/el:20021120

[8] Abid, I., Boudabous, A., Atitallah, A.B. (2015). A new adaptive vector median rational hybrid filter for impulsive noise suppression. In 2015 16th International Conference on Sciences and Techniques of Automatic Control and Computer Engineering (STA), Monastir, Tunisia, pp. 417-421. https://doi.org/10.1109/sta.2015.7505212

[9] Khriji, L., Gabbouj, M., Bernacchia, G., Sicuranza, G.L. (1999). Hardware implementation of the median-rational hybrid filters for colour images. In Proceedings of the IEEE-EURASIP Workshop on Nonlinear Signal and Image Processing NSIP'99, Antalya, Turkey, pp. 124-128.

[10] Boudabous, A., Atitallah, A.B., Khriji, L., Kadionik, P., Masmoudi, N. (2010). HW/SW design-based

implementation of vector median rational hybrid filter. The International Arab Journal of Information Technology, 7(1): 70-74.

[11] Bernacchia, G., Khriji, L., Gabbouj, M., Sicuranza, G.L. (1999). Programmable hardware implementation for the median-rational hybrid filters. In Proceedings 1999 International Conference on Image Processing (Cat. 99CH36348), Kobe, Japan, pp. 182-186. https://doi.org/10.1109/icip.1999.819574

[12] Khriji, L., Gabbouj, M. (2002). Adaptive fuzzy order statistics-rational hybrid filters for color image processing. Fuzzy Sets and Systems, 128(1): 35-46. https://doi.org/10.1016/s0165-0114(01)00181-6

[13] Atitallah, A.B., Kammoun, M., Atitallah, R.B. (2020). An optimized FPGA design of inverse quantization and transform for HEVCdecoding blocks and validation in an SW/HW environment. Turkish Journal of Electrical Engineering and Computer Sciences, 28(3): 1656-1672. https://doi.org/10.3906/elk-1910-122

[14] Boudabous, A., Atitallah, A.B., Khriji, L., Kadionik, P., Masmoudi, N. (2011). FPGA implementation of vector directional distance filter based on HW/SW environment validation. AEU-International Journal of Electronics and Communications, 65(3): 250-257. https://doi.org/10.1016/j.aeue.2010.02.012

[15] Ben Atitallah, A., Abid, I., Boudabous, A., Loukil, H. (2021). A new hardware architecture of the adaptive vector median filter and validation in a hardware/software environment. International Journal of Circuit Theory and Applications, 49(8): 2329-2347. https://doi.org/10.1002/cta.3000

[16] Ben Atitallah, A., Kammoun, M., Ali, K.M., Ben Atitallah, R. (2020). An FPGA comparative study of high-level and low-level combined designs for HEVC intra, inverse quantization, and IDCT/IDST 2D modules. International Journal of Circuit Theory and Applications, 48(8): 1274-1290. https://doi.org/10.1002/cta.2790

[17] Atitallah, A.B., Abid, I. (2020). An efficient FPGA implementation of AVMF filter using high-level synthesis. In 2020 20th International Conference on Sciences and Techniques of Automatic Control and Computer Engineering (STA), Monastir, Tunisia, pp. 82-85. https://doi.org/10.1109/sta50679.2020.9329336

[18] Astola, J., Haavisto, P., Neuvo, Y. (1990). Vector median filters. Proceedings of the IEEE, 78(4): 678-689. https://doi.org/10.1109/5.54807

[19] Boudabous, A., Atitallah, A.B., Kadionik, P., Masmoudi, N. (2007). HW/SW FPGA implementation of vector median filter. In 2007 Ph.D Research in Microelectronics and Electronics Conference, Bordeaux, France, pp. 101-104. https://doi.org/10.1109/rme.2007.4401821

[20] Khriji, L., Gabbouj, M. (1999). Class of multichannel image processing filters. Electronics Letters, 35(4): 285-287. https://doi.org/10.1049/el:19990250

[21] Atitallah, M.A.B., Kachouri, R., Atitallah, A.B., Mnif, H. (2022). An efficient HW/SW design for text extraction from complex color image. CMC-Computers, Materials & Continua, 71(3): 5963-5977. https://doi.org/10.32604/cmc.2022.024345

[22] Kammoun, M., Atitallah, A.B., Ali, K.M., Atitallah, R.B. (2019). Case study of an HEVC decoder application using high-level synthesis: Intraprediction, dequantization, and inverse transform blocks. Journal of Electronic Imaging, 28(3): 033010. https://doi.org/10.1117/1.JEI.28.3.033010

[23] Kammoun, M., Ben Atitallah, A., Ben Atitallah, R., Masmoudi, N. (2017). Design exploration of efficient implementation on SoC heterogeneous platform: HEVC intra prediction application. International Journal of Circuit Theory and Applications, 45(12): 2243-2259. https://doi.org/10.1002/cta.2308

[24] Yalman, Y., Ertürk, I. (2013). A new color image quality measure based on YUV transformation and PSNR for human vision system. Turkish Journal of Electrical Engineering and Computer Sciences, 21(2): 603-612. https://doi.org/10.3906/elk-1111-11