Proceedings of the 4th International Conference on Signal Processing and Machine Learning
DOI: 10.54254/2755-2721/48/20241636

A real time hardware design for bilateral filtering

Minshan Jin

School of Electronic and Information Engineering, Sun Yat-sen University, Guangzhou, China

jinmsh3@mail2.sysu.edu.cn

Abstract. Bilateral filtering is a widely used video image filtering method, but its computational complexity is high, which is not conducive to hardware real-time applications. Therefore, this paper presents a hardware structure of bilateral filtering for real-time applications of high-definition images. Our hardware design mainly includes lookup table structure and parallel structure designs, respectively. Based on the Vivado HLS evaluation, the experimental results show that the hardware structure of this paper can meet the real-time application of bilateral filtering in high-definition video images.

Keywords: Bilateral filtering, Real-time application, Hardware architecture, Lookup table, Parallel structure

1. Introduction

During the process of image acquisition and transmission, noise is inevitably present due to various factors, including thermal noise during acquisition and noise introduced during transmission. Noise has a significant impact on image pixels, which can result in decreased image quality, contrast, and content corruption. Hence, the main objective of image denoising is to effectively remove noise from images. Image denoising involves filtering out noise from an image to enhance its quality

Image denoising involves the proposal and design of numerous image filters, based on a substantial amount of human knowledge in physics. In a recent publication [1], one type of median filter was introduced. This filter utilized two methods to detect noise points and noise density. These methods enabled the filter to detect noise points and estimate noise density. Moreover, it dynamically adjusted the size of the filter window based on the image characteristics. Consequently, the filter can effectively perform adaptive median filtering, successfully reducing noise while preserving image details. Unfortunately, the median filter is not efficient in removing Gaussian noise and may potentially distort graph edges. Another method, proposed in [2], employed a weight based on local variance to achieve effective anisotropic filtering. Additionally, it employed a weighted average to perform region-selective diffusion, which achieved a similar smoothing effect as global filters. It is worth noting that bilateral filtering is a reliable option among numerous image filtering methods, as it effectively enhances image edges and is straightforward to implement.

However, the traditional bilateral filtering may not be suitable for effectively handling significant amounts of noise. In a recent publication [3], a novel algorithm called the Boundary Switching Bilateral Filter was introduced. This algorithm incorporated a method for identifying boundary noise prior to filter application. By incorporating this approach, the accuracy of noise detection is significantly

DOI: 10.54254/2755-2721/48/20241636

enhanced, particularly in high noise scenarios where the ability to dynamically adjust the window size based on noise density proves advantageous. The experimental results obtained using this algorithm surpass those achieved by other switching bilateral filters, particularly when confronted with high standard deviations of noise. In a recent study [4], another novel technique for optimizing the bilateral filter through edge detection has been proposed. This method employed a new variant of bilateral filtering that accentuates the high spatial frequency region within an image, corresponding to the edges, by utilizing a two-dimensional spatial gradient measurement. Furthermore, it effectively mitigated the influence of external edge noise by precisely identifying the original image's edges. Experimental findings have demonstrated a slightly superior suppression effect on salt and pepper noise in comparison to existing methods. In one more another work [5], an innovative adaptive actor-critical bilateral filter has been introduced. This approach integrated a multi-agent algorithm to dynamically adjust the range kernel width in the continuous bilateral filtering process, thereby maximizing image smoothness while preserving the edges even when subjected to slight perturbations.

Although the algorithms have demonstrated varying levels of improvement in the performance of the bilateral filter, they still exhibit high complexity, which challenge a lot for the real-time hardware implementation due to its high computational cost. Therefore, studying hardware-efficient structure designs for the bilateral filter becomes imperative and essential for future applications.

In [6], a specific hardware-friendly technique for denoising grayscale and color images using a bilateral filter was introduced. The values of the register matrix were systematically organized, and potential intensity or coefficient differences were determined in advance within the photometry filter. These values were then stored in a lookup table. To simplify the hardware design, two one-dimensional filters were adopted for gray filtering instead of two-dimensional filters, resulting in a significant reduction in hardware design complexity. In [7], a bilateral grid with variable-sized windows was implemented by introducing a window radius parameter, effectively restraining the growth of hardware resources as the window size increases. A novel solution that facilitates intra image content-adaptive bilateral filtering by modulating range and space sigma through virtual sub-tables was proposed in [8]. This approach reduced both hardware design complexity and performance impact on the original algorithm to a negligible extent.

While there have been efforts made to optimize hardware bilateral filtering, certain structural designs tend to be complex and the implementation process using traditional RTL description statements can be excessively time-consuming, which will become the challenge for fast practical realization of the real time application.

To address this issue, this paper explores the hardware design of bilateral filters using the efficient design method of Vivado HLS. We propose an efficient hardware structure for bilateral filters specifically tailored for real-time processing of high-definition videos. The bilateral filter algorithm is optimized using a lookup table structure and hardware efficiency is enhanced through parallel architectures. Experimental results demonstrate that this method effectively reduces the resource consumption of the bilateral filtering algorithm, thus meeting the requirements for real-time operation for 1920x1080@60fps.

The remainder of this work is organized as follows. Section II provides a further introduction to several related works, followed by the detail's presentation of the proposed hardware architecture in Section III. The experimental results are presented in Section IV, and a summary is drawn in Section V.

2. Related Work

2.1. Bilateral Filter

Bilateral filtering supplements Gaussian filtering by incorporating a range domain weight. It considers both spatial factors and the influence of pixel differences. The weight assigned to the range domain is determined by the disparity in pixel values, with the pixel most like the center pixel receiving a higher weight. Firstly, its two weights are computed as,

DOI: 10.54254/2755-2721/48/20241636

$$G_s = e^{-\left(\frac{||p-q||^2}{2\sigma_s^2}\right)}, \tag{1}$$

$$G_r = e^{-\left(\frac{||I_p-I_q||^2}{2\sigma_r^2}\right)}, \tag{2}$$

where p and q are spatial indexes, I_p and I_q are corresponding pixel values, respectively. G_s is the weight of spatial domain, G_r is the weight of range domain, while σ_s and σ_r are standard deviations, respectively.

Then the filtering result BF is expressed as follow,

$$BF = \frac{1}{W_q}\sum_{p\epsilon s} G_s(p)G_r(p) * I_q = \frac{1}{W_q}\sum_{p\epsilon s} e^{-\left(\frac{||p-q||^2}{2\sigma_s^2}\right)} e^{-\left(\frac{||I_p-I_q||^2}{2\sigma_r^2}\right)} * I_q, \tag{3}$$

where I_q is the input image, W_q is the weight sum of each pixel value in the filtering window, which is used for the normalization of the weight as computed as,

$$W_q = \sum_{p\epsilon s} G_s(p)G_r(p) = \sum_{p\epsilon s} e^{-\left(\frac{||p-q||^2}{2\sigma_s^2}\right)} e^{-\left(\frac{||I_p-I_q||^2}{2\sigma_r^2}\right)}. \tag{4}$$

In flat regions, the filter's pixels exhibit a high degree of similarity, with the spatial weighting factor being the dominant influence on the filtering outcome. Conversely, in edge regions, the values of G_r on one side of the edge greatly outweigh those of the $G_r$values on the other side. Consequently, the impact of pixel weights from the opposite side is minimal, effectively safeguarding the integrity of edge information.

2.2. Edge-Aware Bilateral Filter Method

In study [4], a novel technique for optimizing the bilateral filter through edge detection has been proposed. The suggested filter aims to partially mitigate these artifacts by incorporating a third kernel in the bilateral filter formulations, in addition to the range kernel G_{r0}. The incorporation of the Robert Cross kernel, a prominent edge detection technique, serves to diminish noise impact on outer edges by identifying edges within the original image.

The final weights W_q are shown below,

$$G_{r0} = e^{-\left(\frac{p}{2\sigma_0^2}\right)}, \tag{5}$$

$$W_q = \sum_{p\epsilon s} e^{-\left(\frac{||p-q||^2}{2\sigma_s^2}\right)} e^{-\left(\frac{||I_p-I_q||^2}{2\sigma_r^2}\right)} e^{\left(\frac{p}{2\sigma_0^2}\right)}. \tag{6}$$

The experimental results are shown as Fig.1, it shows the performance of 3 different data sets using various filters as bilateral filter, NLM (Non-local Means filter), anisotropic filter, and their respective PSNR (peak-signal to noise ratio). The proposed method in [4] outperforms the other methods with more PSNR results.

DOI: 10.54254/2755-2721/48/20241636

Figure 1. Experimental results of method proposed in [4].

The proposed method effectively eliminates noise in certain areas where the bilateral filter falls short, resulting in a slightly improved outcome compared to the standard bilateral filter. Additionally, this novel approach overcomes the limitation of gradient inversion encountered in the bilateral filter, as it does not introduce any false edges to the image, and the effect of salt and pepper noise is reduced. Nevertheless, it is important to note that this method does not address the ladder effect observed in bilateral filters, which imparts a cartoonish appearance to images. Furthermore, the inclusion of extra computation including the convolution and division operation in this method will introduce more challenges for real-time hardware design.

Table 1. Experimental results of structure proposed in [7]

	GPU	ICCEE 2008	TIE2014	TIE2018	Design in [7]
FPGA family	N/A	Altera Cyclone II	Xilinx Virtex5	Xilinx Zynq-7000	Xilinx Zynq UltraScale+
w*h	1920*1080	1024*1024	1024*1024	256*256	1920*1080
r	12	1	2	12	12
Other conditions	$\sigma_r = 70$ $\sigma_r = 8$	p = 8	None	σr = 70 N = 9	σr = 70 σr = 8
fclk(Mhz)	N/A	159	220	60	214
fps(fps)	59.88	151.49	52.35	95.60	99.24
Tp(nsec/pixel)	8.05	6.29	18.18	159.61	4.86
Slice	N/A	567	1447	157	1611
LUT	N/A	N/A	N/A	2633	9013
FF	N/A	N/A	N/A	685	7438
DSP	N/A	N/A	29	10	15
BRAM	N/A	N/A	11	157	26.5

2.3. Fully Pipelined Bilateral Grid

In study [7], a real-time image denoising system that uses an FPGA based on the bilateral grid (BG) had been proposed. The bilateral gird (BG) structure is shown as Fig 2. Firstly, the input image pixels p(x, y) are read from DRAM one by one using the AXI bus and DMA. During this process, the input image is converted into a grid by the Grid Converter (GC) for each input pixel. Following this, the Gaussian Filter (GF) blurs the grid after preparing the minimum elements (3 × 3 × 3). The resulting values from this operation are stored in BRAMs (Block RAMs) grid. Finally, the Minimum Elements undergo

DOI: 10.54254/2755-2721/48/20241636

Trilinear Interpolation (TI) after being blurred by the Gaussian Filter. Consequently, for each r line of the input image, r lines of the output image are produced.

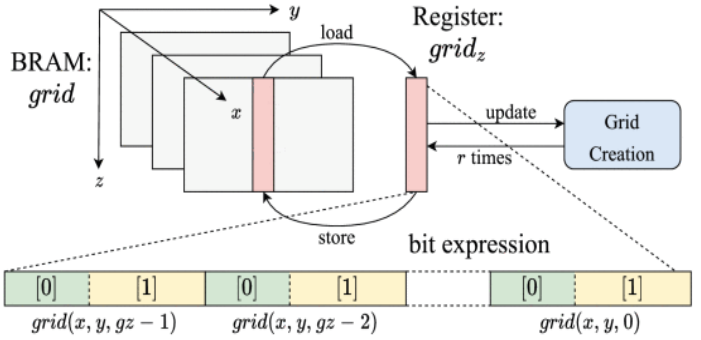


Figure 2. Bilateral grid structure [7].

In addition, this method pipeline three for loops together to form a unified for loop to optimize hardware performance as shown in Fig.3. The direction of the arrow indicates the loading and storage location of the data. The numbers in grid 2D correspond to the numbers in register reg_{GF}.

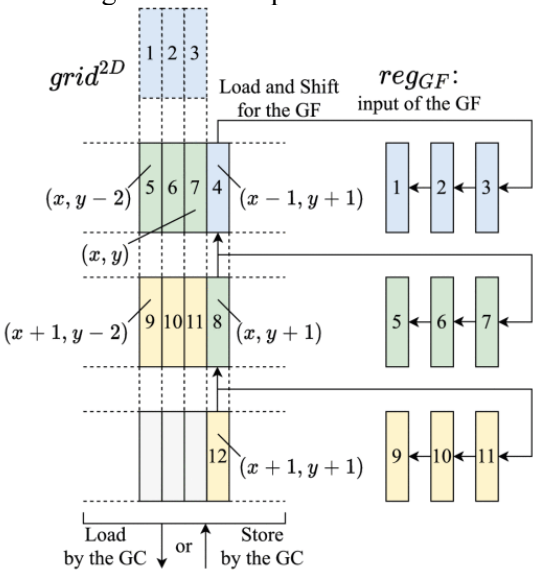


Figure 3. Pipeline Structure of the method in [7].

The experimental results are shown as Table.1, where fclk represents the maximum clock frequency and Tp represents the time consumed per pixel. The results clearly demonstrate the significantly high speed of full high-definition image calculation using the proposed method. Additionally, the consumption of hardware resources remains relatively stable even as the value of r increases. The FGPA pipeline implementation scheme for the proposed BG effectively mitigates any excessive allocation of hardware resources. The practical implementation of the proposed design on an FPGA board establishes its superiority over existing designs in both computational speed and efficient utilization of hardware resources.

2.4. Vivado HLS

Vivado HLS facilitates the translation of a C, C++, or SystemC design into a Register Transfer Level (RTL) implementation. This implementation can be further synthesized and implemented within the programmable logic of a Xilinx FPGA or Zynq chip. Notably, the utilization of Vivado HLS allows for

DOI: 10.54254/2755-2721/48/20241636

the realization of a C design in programmable logic, thereby obviating the need for hardware description languages and instead employing the C language for hardware design.
Vivado HLS enables the creation of hardware description code using high-level programming languages, offering designers the opportunity to work at a higher level of abstraction without the need for direct manipulation of the underlying hardware description language. This approach significantly enhances design efficiency. By incorporating Vivado HLS into the workflow, hardware designs can be swiftly optimized through multiple iterations and debugging. Compared to the manual composition of hardware description languages, Vivado HLS's advanced synthesis process is faster and more adaptable, facilitating easier design iteration. Additionally, equipped with an automated optimization feature, Vivado HLS streamlines the optimization of hardware designs based on predefined performance metrics. This mechanism effectively minimizes the laborious task of manual tuning, resulting in a more efficient hardware implementation.
The design process of Vivado HLS is as follows.

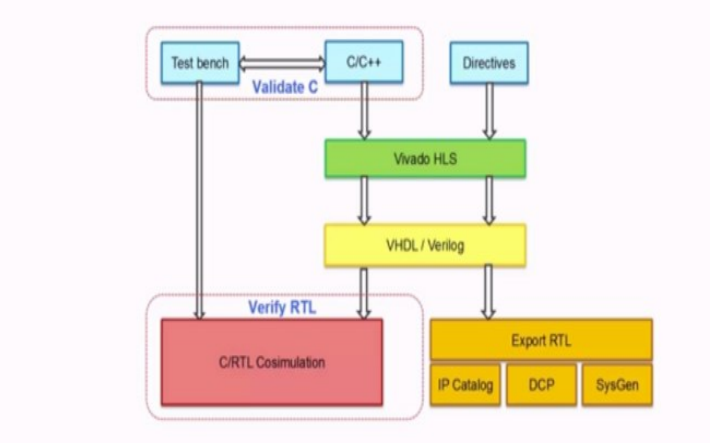


Figure 4. Specific design process of Vivado HLS.

As shown in Fig.4, a test bench will be developed to validate the completed C language code. This whole process will involve importing the code into the test bench, generating the Register Transfer Level (RTL) hardware structure description language. As to validation, the process will conduct the C validation for the algorithm. Subsequently, a co-simulation will be performed by united the C and RTL implementation together to further validate the design's effectiveness and performance.

3. Proposed Hardware Design of Bilateral Filter

Our objective of this paper is to achieve the real time bilateral filter for the resolution of 1920x1080 with a frame rate of 60 fps at the frequency 300MHz. Obviously, it is hard to process the bilateral filter at the whole frame level once due to the limitation storage of the hardware designs. Therefore, we propose a block level bilateral filtering hardware structure. What's more within the realm of bilateral filtering data processing, numerous multiplication and exponential operations arise, considerably prolonging operational time. Additionally, within each block, the same calculation must be executed for every pixel in the 3*3 window, resulting in multiple loops that necessitate additional calculations. All above will challenge the hardware design of bilateral filter.
To accomplish this, our algorithm employs a 16x16 pixel filter unit and a 3x3 filter core. The following N is the number of cycles required for a certain block with size $S * S$,where m is the hardware operation frequency, W and H are the width and height of the image, respectively, and S is the size of block. Based on the calculations, we need to realize the cycle number of 617 to achieve the goal of bilateral filtering at 1920x1080@60fps.

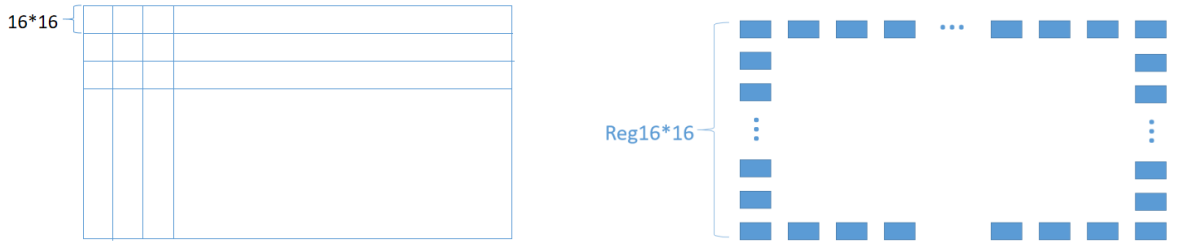
$$N = m/(W * H * fps/S^2), \quad (7)$$

DOI: 10.54254/2755-2721/48/20241636

The hardware architecture of bilateral filtering discussed in this paper comprises three main components: block partitioning design, lookup table design, and parallel structure design as presented below.

3.1. Block Partitioning

For a certain image with width W and height H, we divvied it into several non-overlapped blocks as shown in Fig. 5(a). To accomplish the block-level objective, we partition the image into numerous 16*16 blocks in the horizontal and vertical directions, respectively.



(a) Schematic Diagram of Blocks (b) Registers of a 16*16 block

Figure 5. Block partition and register method.

As shown in Fig.5 (a) and (b), we divide the 1920 * 1080 image into multiple 16 * 16 blocks with corresponding registers, and the pixel values are stored in 16 * 16 independent registers. When filtering the pixels in this block, we directly fetch the data from the registers immediately without any data dependency.

3.2. Lookup Table

During the calculation process of bilateral filtering, the computational complexity of variable W_q in Eq. (4) is significantly high. This is due to the involvement of operations such as square sum division, exponential calculation, and summation. Consequently, these complex calculations substantially increase the runtime of the hardware. Therefore, we design the coefficients lookup table structure to calculate and store the possible spatial and range weights to optimize hardware speed.

The spatial weight in the bilateral filter remains constant for each coordinate, while the pixel weight is also fixed for every pixel difference, enabling their pre-calculation and storage. To obtain the respective spatial weight, each potential coordinate is input into Eq. (1). The images utilized in the experiment are 8-bit, with pixel values spanning from 0 to 255. By substituting each feasible pixel value into Eq. (1), the corresponding pixel weight is obtained. The weights contained within the two tables are then multiplied separately, resulting in the production of a secondary lookup table comprising products of all possible weights. The value within the coordinate lookup table is modified to match the corresponding pixel number, and together with the secondary lookup table, addresses the pixel value. Therefore, we skip the complicated multiplication and exponential operations in the filtering process. The final lookup table of the final weight W_q is partially shown as Table.2, which can be utilized as referenced immediately to accomplish the bilateral filtering of each current pixel.

Table 2. Lookup Table for the weight coefficients of W_q

$$W_q = \begin{bmatrix} 0.99 & 0.98 & 0.97 & \dots \\ 0.94 & 0.91 & 0.87 & \dots \\ 0.82 & 0.77 & 0.71 & \dots \\ \dots & \dots & \dots & \dots \end{bmatrix}$$

DOI: 10.54254/2755-2721/48/20241636

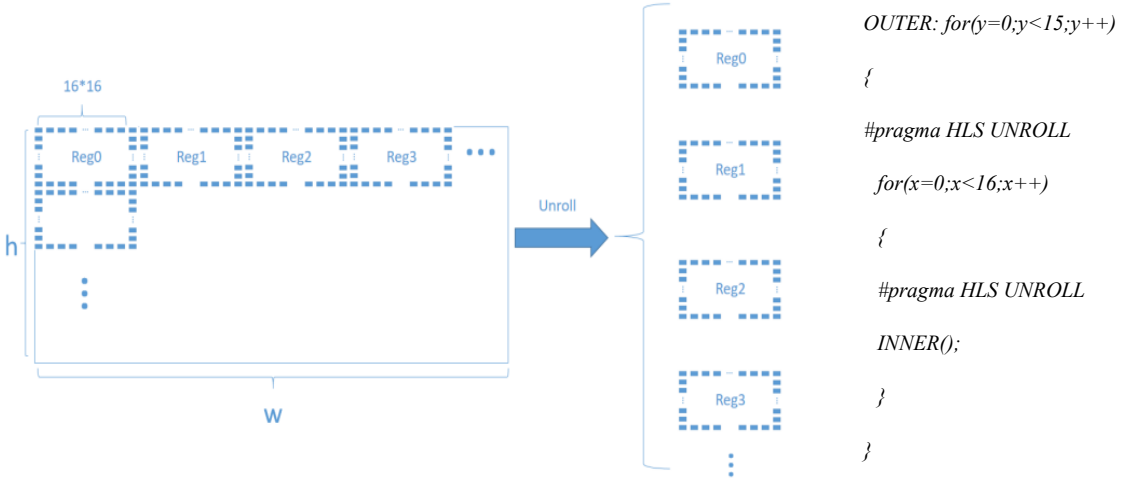
3.3. Parallel Structure

Although we have addressed issues like multiplication and exponential operations, which are unsuitable for hardware computing, by implementing lookup tables, there remains a substantial number of pixels that require computation. Serial processing proves inadequate for achieving real-time bilateral filtering of high-definition videos. Consequently, we employ parallel concepts to enhance the hardware execution speed of bilateral filtering.

Our algorithm has four parts, with outer parts to divide a frame of image into 16 * 16 blocks, and the inner parts to accomplish the bilateral filtering process in a 3 * 3 window.

We utilized Vivado HLS to unroll the algorithm implementation into multiple instruction streams. The execution of block partitioning and window partitioning instruction streams are conducted in parallel to enhance performance. The execution process of each instruction is divided into several stages, enabling different instructions to be executed simultaneously without any mutual interference. This enables the processor to execute multiple instructions within a single clock cycle, thereby significantly improving hardware efficiency.

Fig.6 (a) demonstrates the parallel structure employed after performing the unroll operation on the outer components. This parallel structure facilitates the simultaneous splitting of all register matrices. The implementation process is briefly shown as in Fig.6 (b) by Pseudo code I. In contrast, the original serial structure requires sequential execution of operations for each block. By unroll operation, we can now concurrently process the pixels within these blocks.



(a) Parallel structure of block

```
OUTER: for(y=0;y<15;y++)
{
#pragma HLS UNROLL
 for(x=0;x<16;x++)
 {
 #pragma HLS UNROLL
 INNER();
 }
}
```

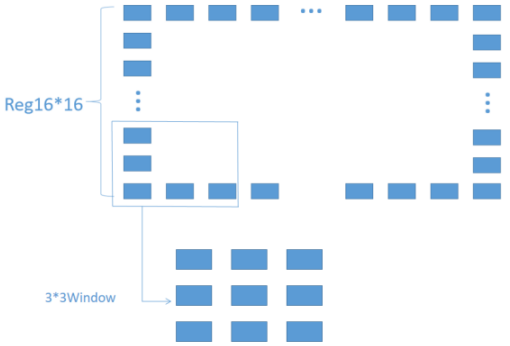
(b) **Pseudo code I:** Pseudo code of block partition (outer parts)

Figure 6. Parallel structure and pseudo code of 16x16 block level (outer parts).

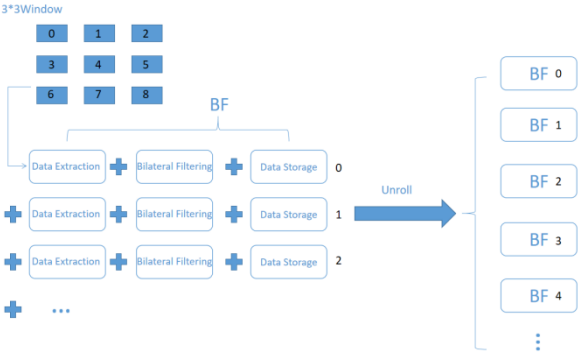
Before performing the unroll operation, each pixel operates sequentially. This implies that each pixel must wait for the preceding pixel to complete the filtering process before commencing its own filtering. It becomes apparent that such a structure does not allow for achieving real time hardware design. Therefore, we design a novel structure for parallel implementation as depicted in Fig. 7.

As shown in Fig.7 (a) and Fig.7 (b), when performing bilateral filtering in each block, we select a 3 * 3 window for each pixel, and perform data extraction, data calculation (BF), and data storage operations for each pixel in the window. The implementation process is briefly shown by Pseudo code II as shown in Fig.7 (c). After executing the unroll operation, each pixel can concurrently carry out the operations. Furthermore, through the implementation of the two unroll operations mentioned previously, we can effectively filter the pixels in a single frame of an image simultaneously, thereby meeting the demands of real-time hardware design.

DOI: 10.54254/2755-2721/48/20241636



(a)3*3 window in block (inner parts)



(b) Parallel structure of BF(inner parts)

INNER ():

for(k=0;k<3;k++)

{

#pragma HLS UNROLL

for(l=0;l<3;l++)

{

#pragma HLS UNROLL

Bilateral_filter();

}

}

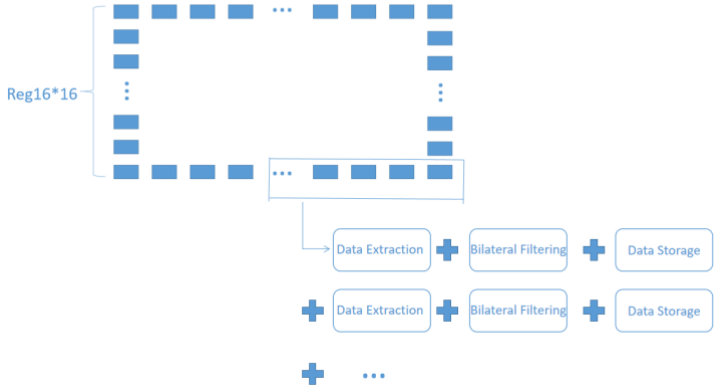
(c) Pseudo code II: Pseudo code of window (inner parts)

Figure 7. Parallel structure and pseudo-code of normal 16x16 level inner window

To mitigate excessive utilization of hardware resources, the serial structure is implemented for the final few pixels within a block. As depicted in Fig.8(a), these pixels undergo sequential data extraction,

DOI: 10.54254/2755-2721/48/20241636

data calculation, and data storage operations. Due to only a subset of pixels are remained for each block 16x16 to the end, we directly employ the serial structure to achieve the cycle goal while conserving hardware resources. The following pseudocode outlines the implementation of the serial structure section. The implementation process is briefly shown by Pseudo code III as shown in Fig.8 (b).



(a) Serial structure of BF

```
for(y=15;y<16;y++)
{
  for(x=0;x<6;x++)
  {
  #pragma HLS UNROLL
       INNER ();
    }
  }
  for(x=6;x<16;x++)
  {
  INNER ();
  }
}
```

(b) Pseudo code III :Pseudo code of serial structure

Figure 8. Serial structure and pseudo code of BF.

3.4. Experimental Results

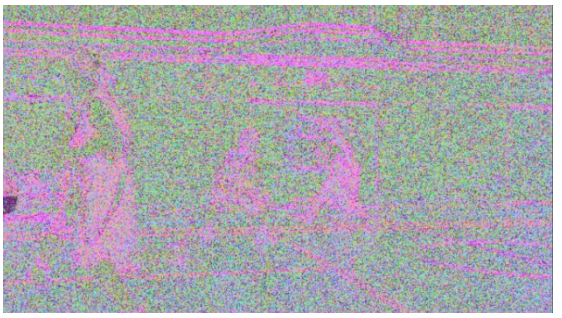
Our objective of this work is to achieve the real time bilateral filter for the resolution of 1920x1080 with a frame rate of 60 fps at the frequency 300MHz. To achieve this goal, it is required to realize the cycle number of 617 for the block with size of 16*16 as mentioned before. We employ a video sequence BasketballPass with 50 frames for bilateral method validation and conduct the hardware architecture design experiment on Vivado HLS 2019.2 with the target device of xczu7ev-ffv1517-1-i and the product family of zynquplus.

(a) Original image

(b) Filtered image (without using lookup table)

DOI: 10.54254/2755-2721/48/20241636

(c) Filtered image (using lookup table)



(d) Images of differences after bilateral filter

Figure 9. Image filtering result comparison.

Firstly, the filtering results are compared to shown the effectiveness of our lookup table based bilateral filter as shown in Fig. 9. Fig.9 (a) presents the unfiltered images, Fig.9 (b) shows the filtered image without using lookup table and Fig.9 (c) depicts the filtered image using lookup tale. It's evident that the filter quality of the image is almost constant before and after using the lookup table. Fig.9(d) illustrates the dissimilarities between the original and lookup table-based method. It is evident from Fig.9(d) that we have successfully applied bilateral filtering to the image.

Secondly, Table.3 shows the number of cycles before and after optimization. It is evident that optimizing the lookup table and parallel structure leads to a substantial reduction in cycle count and significant improvement in hardware speed, successfully achieving our objectives, which should be smaller than 617 cycles. Finally, Table.4 shows the resource usage before and after optimization. The lookup table structure and parallel structure consume more hardware resources to improve hardware performance. For example, the LUT utilization is more than original due to the our substantial unroll and partition operations for registers. Also, the DSP will be more due to many multiplication operations at same time.

Table 3. Hardware Cycles Speed Up Results

	Before optimization	After optimization
Cycles	2912385	573

Table 4. Hardware Resource usage Comparison

(a) Resource usage before optimization

Name	BRAM	DSP	FF	LUT
DSP	-	-	-	-
Expression	-	-	0	947
FIFO	-	-		
Instance	68	190	19539	27330
Memory	-	-	-	-
Multiplexer	-	-	-	616
Register	-	-	1610	-
Total	68	190	21149	28893
Available	624		460800	230400
Utilization	10	10	4	12

DOI: 10.54254/2755-2721/48/20241636

(b) Resource usage after optimization

Name	BRAM	DSP	FF	LUT
DSP	-	-	-	-
Expression	-	-	0	120954
FIFO	-	-	-	-
Instance	-	1229	188703	355393
Memory	861	-	8	1
Multiplexer	-	-	-	41801
Register	-	-	41819	-
Total	861	1229	230530	518149
Available	624	1728	460800	230400
Utilization	137	71	50	224

4. Conclusion

In this paper, we propose a hardware structure of bilateral filtering for high-definition real-time applications. Our structure mainly includes two level lookup table structure and parallel structure. The experimental results show that the hardware structure of this paper can satisfy the real-time filtering of high-definition video images. In the future, we will further combine algorithms and hardware to further study more efficient bilateral filtering algorithms and hardware architectures.

References

[1] J. Tang, Y. Wang, W. Cao and J. Yang, "Improved Adaptive Median Filtering for Structured Light Image Denoising," 2019 7th International Conference on Information, Communication and Networks (ICICN), Macao, China, 2019, pp. 146-149, doi: 10.1109/ICICN.2019.8834974.

[2] C. N. Ochotorena and Y. Yamashita, "Anisotropic Guided Filtering," in IEEE Transactions on Image Processing, vol. 29, pp. 1397-1412, 2020, doi: 10.1109/TIP.2019.2941326.

[3] K. S. Rani and R. V. S. Satyanarayana, "Image denoising using boundary discriminated switching bilateral filter with highly corrupted universal noise," 2017 International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS), Chennai, India, 2017, pp. 1515-1521, doi: 10.1109/ICECDS.2017.8389699.

[4] A. Kaur, H. Singh and D. Arora, "An efficient approach for image denoising based on edge-aware bilateral filter," 2017 4th International Conference on Signal Processing,Computing and Control (ISPCC), Solan, India, 2017, pp. 56-61, doi: 10.1109/ISPCC.2017.8269649.

[5] B. -H. Chen, H. -Y. Cheng and J. -L. Yin, "Adaptive Actor-Critic Bilateral Filter," ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Singapore, Singapore, 2022, pp. 1675-1679, doi: 10.1109/ICASSP43922.2022.9746631.

[6] C. S. Chandni and R. Pushpakumari, "Reduced hardware architecture of bilateral filter for real time image denoising," 2017 International Conference on Intelligent Computing, Instrumentation and Control Technologies (ICICICT), Kerala, India, 2017, pp. 769-774, doi: 10.1109/ICICICT1.2017.8342661.

[7] N. Hashimoto and S. Takamaeda-Yamazaki, "An FPGA-Based Fully Pipelined Bilateral Grid for Real-Time Image Denoising," 2021 31st International Conference on Field-Programmable Logic and Applications (FPL), Dresden, Germany, 2021, pp. 167-173, doi: 10.1109/FPL53798.2021.00035.

[8] M. Mody, R. Allu, J. Villarreal, W. Wallace, N. Nandan and A. Baranwal, "High Throughput VLSI Architecture for Bilateral Filtering in Computer Vision," 2022 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT), Bangalore, India, 2022, pp. 1-4, doi: 10.1109/CONECCT55679.2022.9865827