Received February 10, 2020, accepted March 19, 2020, date of publication March 31, 2020, date of current version April 16, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2984688

A Low-Cost VLSI Architecture of the Bilateral Filter for Real-Time Image Denoising

CHIH-YUAN LIEN[1], CHI-HUAN TANG[2], PEI-YIN CHEN[2], (Member, IEEE), YAO-TSUNG KUO[2], AND YUE-LING DENG[2]
[1]Department of Electronic Engineering, National Kaohsiung University of Sciences and Technology, Kaohsiung 80778, Taiwan
[2]Department of Computer Science and Information Engineering, National Cheng Kung University, Tainan 70101, Taiwan

Corresponding author: Pei-Yin Chen (pychen@mail.ncku.edu.tw)

This work was supported in part by the Ministry of Science and Technology, Taiwan, under Grant MOST 108-2221-E-006 -096 -MY3.

ABSTRACT In this paper, a low-cost hardware architecture of the bilateral filter for real-time image processing is proposed. Based on the techniques of distance-oriented grouping and hardware resource sharing, the usage of multipliers can decrease 48% as compared to the previous approach. Besides, an efficient quantization method is applied to reduce the size of required look up tables. The experimental results show that the proposed architecture is cost efficient while maintaining the same image quality, frame rate and working clock frequency.

INDEX TERMS Bilateral filter, image processing, noise reduction, real-time processing.

I. INTRODUCTION

Image denoising is an important technique in image processing. Among the many denoising technologies, bilateral filtering [1] is well-known and remarkable in image processing because it can reduce noise while preserving details of the image. Bilateral filter has been widely used in many applications, such as stereo matching [2][3], denoising [4][5], specular highlight removal [6][7] and detail-enhanced fusion [8]. These applications have been implemented for enhancing image quality on various devices, including digital still cameras, smartphone and surveillance camera. However, bilateral filtering requires intensive computations, which significantly increase computation time for processing high resolution image. Therefore, the hardware implementation is unavoidable to accelerate the speed for real-time applications.

To improve the speed of bilateral filtering, several previous works [9]–[13] in hardware architecture have been implemented and presented on field-programmable gate arrays (FPGAs). In [9], the authors attempted to decrease the resource demand by sorting data into several groups and proposed a less expensive hardware architecture than other designs [10]-[12]. In this paper, a low-cost and real-time very large scale integrated (VLSI) architecture of a bilateral filter that does not adversely affect the quality of the image is proposed. To achieve this goal, distance-oriented grouping, resource sharing and size reduction of look-up tables (LUT) are applied. Compared with [9], the proposed design reduces approximately 48% of the multipliers, 95% of the total LUT size, and 27% of logic slices for Xilinx Virtex XC5VLX50-1 FPGA without influencing the frame rate, working clock frequency, or the quality of image.

The associate editor coordinating the review of this manuscript and approving it for publication was Larbi Boubchir.

II. OVERVIEW OF BILATERAL FILTER

The bilateral filter [1] comprises two components: geometric and photometric. The geometric component, also known as the domain filter, is implemented using a spatial filter and is often formulated as a low-pass filter. Through averaging the nearby pixel value, the domain filter can complete entire denoising process. However, linear averaging overly blurs the edges of objects during denoising. To preserve the edges, a photometric component called range filter is employed to average similar pixel values. As a nonlinear component, the range filter is programmed to disregard the pixel whose value diverges from the value of the center pixel in the filter window by a specified amount, regardless of its position.

The bilateral filter with the domain and range filters is described as follows:

$$\hat{f}\,(x,y) = \frac{K\,(x,y)}{N\,(x,y)} = \frac{\sum_{(s,t)\in\Omega} f\,(x,y)\,W_d\,(s,t)\,W_r\,(s,t)}{N\,(x,y)} \tag{1}$$

where $\hat{f}$ is the filtered image, (x,y) denotes the coordinate of the current pixel to be filtered, $K(x,y)$ is the non-normalized kernel result, $N(x,y)$ is the normalization term, f is the

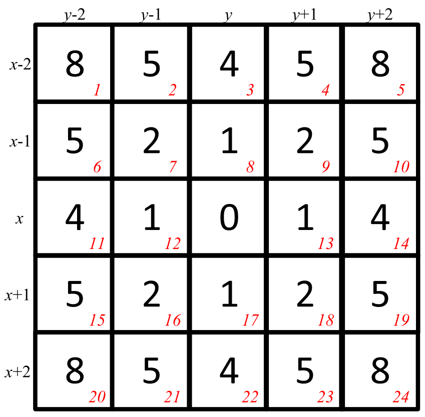

FIGURE 1. Distance-oriented grouping of window size 5 × 5. Note: The black numbers in the center of each square represent the Euclidean distances between the current pixel and the corresponding pixels and the red numbers in the lower right mean the 24 neighboring pixels of (x, y).

input image to be filtered, Ω is the filter window centered in (x, y), (s, t) denotes the spatially neighboring pixel coordinate of current pixel in Ω, $W_d(s, t)$ is the weight of geometric component and $W_r(s, t)$ represents the weight of photometric component.

Taking Gaussian noise into account, $W_d(s, t)$ and $W_r(s, t)$ are represented as the expressions (2) and (3), respectively, both of which are derived from the Gaussian curve:

$$W_d(s,t) = \exp(-\frac{D(f\ (x,y)\ , f(s,t))^2}{2\sigma_d^2}), \tag{2}$$

$$W_r(s,t) = \exp(-\frac{\delta(f\ (x,y)\ , f(s,t))^2}{2\sigma_r^2}), \tag{3}$$

where $D(f(x, y), f(s, t))$ and $\delta(f(x, y), f(s, t))$ represent the Euclidean distance and intensity difference respectively between the current pixel $f(x, y)$ and its neighborhood pixel $f(s, t)$, and σ_d and σ_r regulate the width of the Gaussian curve assigned to $W_d(s, t)$ and $W_r(s, t)$, respectively. Obviously, the photometric component cannot be calculated in advance because of the operation rule, hence the division used for normalization (see (3)) is unavoidable.

Subsequently, normalization term is performed with $N(s, t)$ to ensure that the range of the filter images does not exceed the specified limits because of filtering:

$$N\ (x,y) = \sum_{(s,t)\in\Omega} W_d\ (s,t)\ W_r\ (s,t). \tag{4}$$

III. PROPOSED METHOD

For real-time applications, hardware implementation of bilateral filtering is necessary. Several bilateral hardware designs [9]–[12] have been presented and implemented on field-programmable gate arrays (FPGAs). The authors of [9] arranged the input data into groups and proposed a kernel-based design to process the entire 5 × 5 filter window at one pixel clock cycle. Fig. 1 shows the Euclidean distances $D(f(x, y), f(s, t))$ of the current pixel at coordinate (x, y) and its 24 neighboring pixels in a 5 × 5 filter window. In [9], they

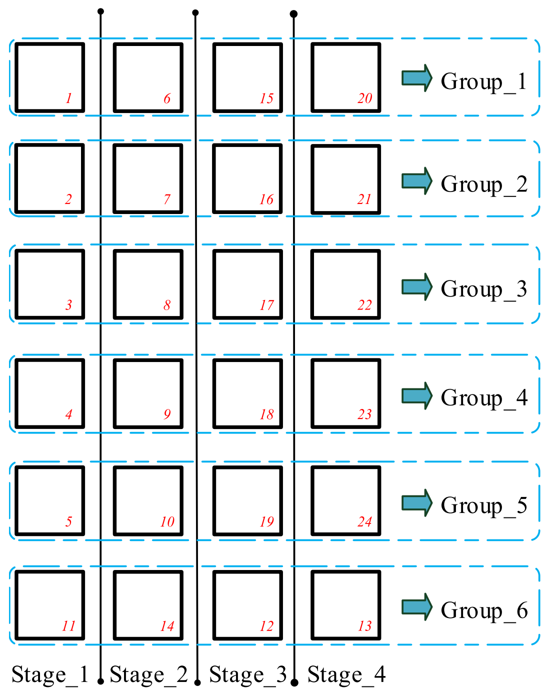

FIGURE 2. Position-oriented grouping of window size 5 × 5 from [9]. The red numbers in the lower right is referred to Fig. 1.

TABLE 1. Processes and number of multipliers in [9].

Step	Process	Input	Output	Multiplier	Total multiplier	Pixel Latency
1	Photometric Filtering	$f(s,t), W_r(s,t)$	$F_r(s,t) = f(s,t) \times W_r(s,t)$ $(s,t) \in \Omega$	7	23	4
2	Kernel Result Calculation	$F_r(s,t), W_d(s,t)$	$K(x,y) = \sum_{(s,t)\in\Omega} F_r(s,t) \times W_d(s,t)$	8		
	Normalization Term Calculation	$W_d(s,t), W_r(s,t)$	$N(x,y) = \sum_{(s,t)\in\Omega} W_d(s,t) \times W_r(s,t)$	8		
3	Normalization	$K(x,y), N(x,y)$	$\hat{f}(x,y) = \frac{K(x,y)}{N(x,y)}$	0		

adopted position-oriented grouping to divide the input data into six groups as shown in Fig. 2. Then, three processing steps are employed to implement bilateral filtering (see (1)) on the six groups of data in parallel. Table 1 lists the detailed processing steps, input and output of each step, number of required multipliers and pixel latency of [9].

Although [9] achieves better hardware resource utilization than [10]–[12], we find that their kernel-based structure can be improved further. By using distance-oriented grouping and resource sharing, we proposed a lower cost hardware in this paper. Besides, the sizes of LUTs used in [9] are still larger, so we applied an efficient quantization method to reduce the sizes of required LUTs. The details of the proposed method are described in the following.

A. DISTANCE-ORIENTED GROUPING

Unlike [9] which adopts the position-oriented grouping shown in Fig. 2, we divide the input data in a 5 × 5 filter window into six groups according to their Euclidean distances

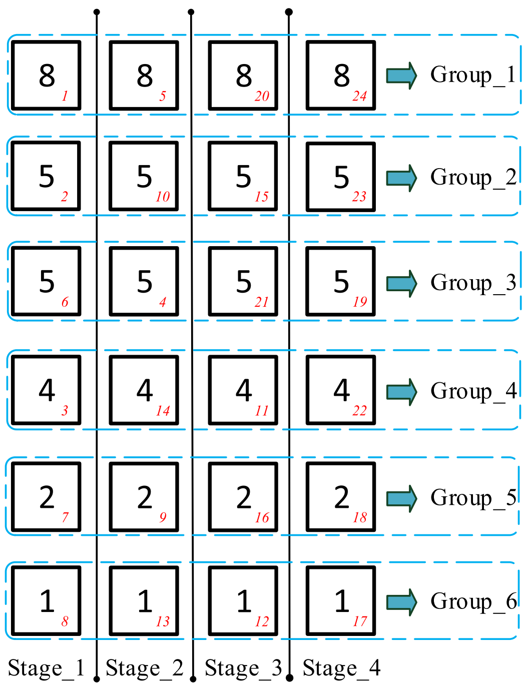
8 1, 8 5, 8 20, 8 24 Group_1
5 2, 5 10, 5 15, 5 23 Group_2
5 6, 5 4, 5 21, 5 19 Group_3
4 3, 4 14, 4 11, 4 22 Group_4
2 7, 2 9, 2 16, 2 18 Group_5
1 8, 1 13, 1 12, 1 17 Group_6
Stage_1
Stage_2
Stage_3
Stage_4

FIGURE 3. Proposed distance-oriented grouping of window size 5 × 5.

from the center pixel. Those pixels having the same Euclidean distances from the center pixel are assigned to one group. Fig. 3 shows the distance-oriented grouping results of input data in our design. Obviously, the number of pixels with distance 5 is double than that of other groups. Thus the eight pixels with distance 5 are separated into two groups to achieve better hardware parallel computation.

After grouping, six hardware modules are used to process the six groups of input data in parallel and the 5×5 filter window can be processed completely after four stages. In other words, the hardware module used for the first group will process input data at coordinates $(x-2, y-2)$, $(x-2, y+2)$, $(x+2, y-2)$ and $(x+2, y+2)$ in sequence. Compared with [9], the proposed distance-oriented grouping can reduce the required adders and selectors.

B. RESOURCE SHARING

To implement bilateral filtering (see (1)), [9] perform the photometric filtering in the first step as mentioned in Table 1. Then, they calculate the kernel result and normalization term concurrently. In the final step, the normalized result is generated.

After careful studying, we discover that the processing steps adopted in [9] require some redundant hardware resources. Observing the numerator and denominator of (1) and (4), we realize that they have the same constituent factor. This factor represents the particular coefficient weight for each pixel in the filter window and can be described as follow:

$$W(s,t) = \begin{cases} constant, & (s,t) \text{ is the central pixel} \\ W_d(s,t)\, W_r(s,t), & otherwise \end{cases} \tag{5}$$

TABLE 2. Processes and number of multipliers in the proposed design.

Step	Process	Input	Output	Multiplier	Total multiplier	Pixel Latency
1	Weight Calculation	$W_d(s,t), W_r(s,t)$	$W(s,t) = W_d(s,t) \times W_r(s,t)$ $(s,t) \in \Omega$	6	12	4
2	Kernel Result Calculation	$f(s,t), W(s,t)$	$K(x,y) = \sum_{(s,t)\in\Omega} f_r(s,t) \times W(s,t)$	6		
	Normalization Term Calculation	$W(s,t)$	$N(x,y) = \sum_{(s,t)\in\Omega} W(s,t)$	0		
3	Normalization	$K(x,y), N(x,y)$	$\hat{f}(x,y) = \frac{K(x,y)}{N(x,y)}$	0		

where $W(s,t)$ is the combined weight of geometric and photometric weights. Obviously, we can reorganize the calculation order of bilateral filtering to achieve resource sharing and reduce the required hardware cost. If $W(s,t)$ is calculated at the beginning, then (1) can be derived by including (5) and rewritten as

$$\hat{f}(\mathrm{x}, \mathrm{y}) = \frac{\sum_{(s,t)\in\Omega} f(s,t)\, W(s,t)}{\sum_{(s,t)\in\Omega} W(s,t)}. \tag{6}$$

In other words, we adopt weight-prior computation to perform bilateral filtering in order to reduce the number of multipliers required for weight calculation. Table 2 lists the detailed processing steps, input and output of each step, number of required multipliers and pixel latency of our design. Evidently, the proposed design reduces the number of multipliers by approximately 48% compared with [9].

C. LUT

Observing (2), we know Euclidean distances $D(f(x,y), f(s,t))$ in the filter window are fixed, so we can calculate the geometric component $W_d(s,t)$ in advance and save those values in the circuit. Thus, no exponential operation is needed to calculate $W_d(s,t)$. However, the exponential operation used to obtain the photometric component $W_r(s,t)$ (see (3)) is unavoidable. To avoid intensive calculations required for the exponential operation, LUT method that records all precalculated weight values is adopted to obtain the exponential result.

Observing (3), we know that the intensity difference $\delta(f(x,y), f(s,t))$ can be used as the input value of LUT to obtain the corresponding $W_r(s,t)$. Because the intensity difference can range from 0 to 255, a LUT with 256 entries is needed for storing all possible values. However, the observation of the possible $W_r(s,t)$ values reveals that some values are close to zero. These values influence the result of bilateral filter slightly, so they can be removed from the LUT to reduce LUT size. After that, the remaining values can be classified into several groups: similar values are assumed to be the same value to further reduce LUT size. Thus, (3) can be rewritten as follows:

$$W_r(s,t) = \begin{cases} 0, & \delta < LUT_{\sigma_r}(0) \\ LUT_{W_r}(\sigma_r, index), & LUT_{\sigma_r}(index) \le \delta \\ & \le LUT_{\sigma_r}(index+1) \end{cases} \tag{7}$$

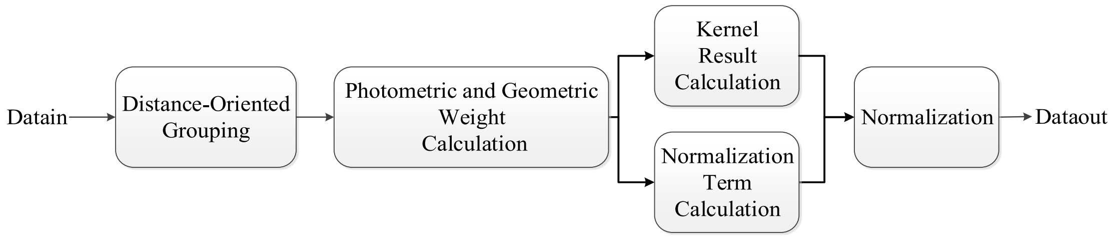
Datain
Distance-Oriented Grouping
Photometric and Geometric Weight Calculation
Kernel Result Calculation
Normalization Term Calculation
Normalization
Dataout

FIGURE 4. Block diagram of proposed VLSI architecture.

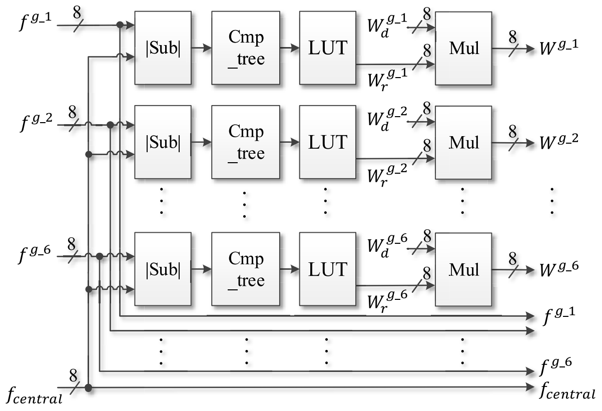

FIGURE 5. Hardware architecture of combined photometric and geometric weight calculation.

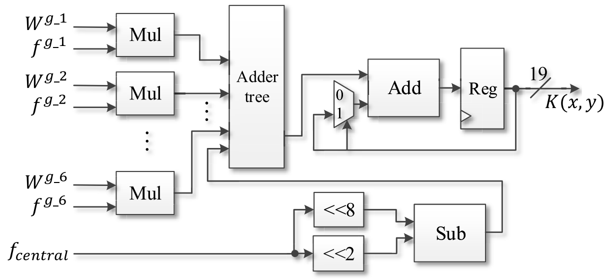

FIGURE 6. Hardware architecture of kernel result calculation.

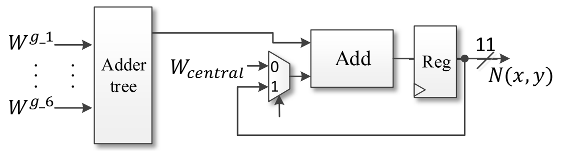

FIGURE 7. Hardware architecture of normalization term calculation.

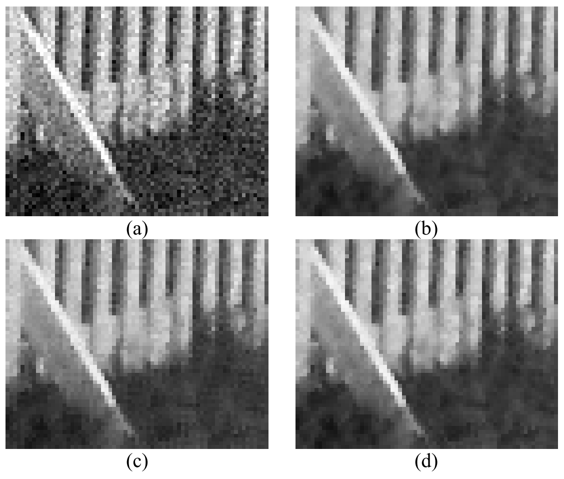

FIGURE 8. Result images for Lighthouse: (a) noisy image with standard deviation 20; (b) filtered using software-oriented bilateral filter; (c) filtered using [9]; (d) filtered using the proposed design.

where LUT_{w_r} stores the exponential value, LUT_{σ_r} stores the boundary of each quantization interval, and an *index* is used to obtain the correct value of the weight. According to our extensive experimental results, we found that twelve groups are enough to achieve comparable filtering performance. Hence, the LUT size of certain σ_r is set to 12. In other words, each LUT adopts 12 entries in our design. Compared with [9] which requires 256 entries in each LUT, our design can reduce the size of LUT by 95%.

IV. IMPLEMENTATION AND EXPERIMENTAL RESULT

Like [9], we also choose a 5×5 filter window for VLSI implementation due to the tradeoff between high noise reduction and low blurring effect. Fig. 4 shows the detailed description of the proposed VLSI architecture. The combined photometric and geometric weight calculation circuit which implements (5) is presented in Fig. 5. Fig. 6 and 7 present the hardware architecture of kernel result and normalization term calculation, respectively. The superscript g in Figs. 5–7 indicates the corresponding data group. In the last step, the normalization module computes the output data (Dataout) and completes the filtering. The proposed design is implemented with a 33-stage pipelined architecture to achieve the speed required for real-time bilateral filtering applications.

Our VLSI architecture is implemented by using the Verilog hardware description language. Then, Mentor Modelsim is used for functional simulation. Finally, Xilinx ISE 14.7 WebPACK version is employed for FPGA realization. Table 3 lists the implementation results of our design and two previous designs [9], [13]. Note that [9] is for grayscale images. However, [13] is for the color image. Therefore, the number of occupied slices usage from [9] is multiplied three times to make a fair comparison. Obviously, the proposed design achieves faster speed and lower cost than [9] and [13]. The authors of [9] have demonstrated that their architecture is better than those previous designs [10]–[12]. Thus, we can conclude that our design outperforms those previous designs [9]–[13]. Table 4 lists the hardware resource

TABLE 3. Implementation results of [9], [13] and the proposed design.

	Window size	Max. clock frequency (MHz)	Maximum frame[1] rate(fps)	Clock cycles/output pixel	Number of occupied slices	Number of Slice LUTs	Number of fully used LUT-FF pairs	Number of bonded IOBs	Number of DSP48Es
[9]	5×5	220	52.45	4	3180	49685	0	832	115
[13]	5×5	N/A	N/A	N/A	N/A	19717	0	832	65
Proposed	5×5	236.697	56.43	4	2304	5142	1782	69	36

[1]Frame size = 1024×1024
N/A, not available.

TABLE 4. Cost of different window sizes for the proposed design.

Window size	Number of grouping	Number of Adder	Number of Multipliers	Clock cycles/output pixel	Max. clock frequency (*MHz*)	Pipeline stage	Throughput(*pixel/s*)
3×3	2	4	4	4	236.697	32	59171146.1
5×5	6	12	12	4	236.697	33	59171132.0
7×7	12	24	24	4	236.697	34	59171117.9
9×9	20	40	40	4	236.697	35	59171103.8

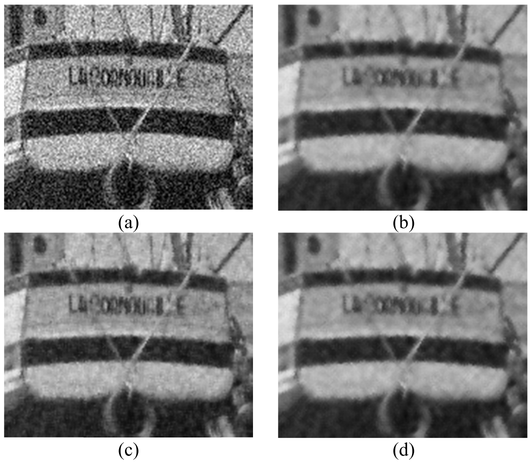

FIGURE 9. Result images for Boat: (a) noisy image with standard deviation 30; (b) filtered using software-oriented bilateral filter; (c) filtered using [9]; (d) filtered using the proposed design.

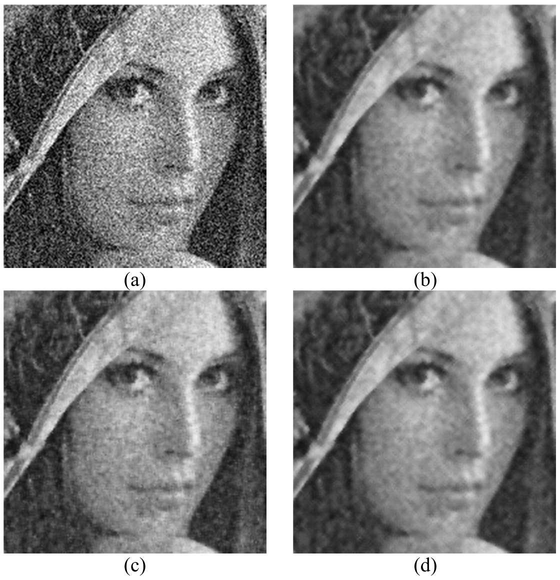

FIGURE 10. Result images for Lena: (a) noisy image with standard deviation 40; (b) filtered using software-oriented bilateral filter; (c) filtered using [9]; (d) filtered using the proposed design.

utilization, speed and throughput of the proposed design when different sizes of filter window are adopted. The hardware resource utilization will increase along with the window size, and the clock frequency and throughput remain almost the same.

Furthermore, to compare the image quality of the proposed design and [9], a variety of simulations were executed on ten well-known 512 × 512 grayscale test images: Lena, Lighthouse, Boat, Couple, Goldhill, Peppers, Plane, Barbara, Bridge, and Lake. In the simulations, the images were corrupted by additive white Gaussian noise (AWGN) with different standard deviations $\sigma_{noise} = [10, 20, 30, 40, 50, 60]$. The filter parameters $\sigma_d = 1$ and $\sigma_r = 3 \times \sigma_{noise}$ were chosen for the photometric and geometric weights, respectively. To illustrate the quality of the reconstructed images, both peak signal-to-noise ratio (PSNR) and structural similarity (SSIM) [14] was employed. PSNR is calculated as

$$\mathrm{MSE} = \frac{1}{xy}\sum_{j=0}^{y-1}\sum_{i=0}^{x-1}\left(I(i,j) - \bar{I}(i,j)\right)^2, \tag{8}$$

$$\mathrm{PSNR} = 20\log_{10}\left(\frac{255}{\sqrt{MSE}}\right), \tag{9}$$

where the size of original grayscale image I is $x \times y$, $\bar{I}$ represents the reconstructed image, MSE represents the mean square error between I and $\bar{I}$, and $I(i,j)$ and $\bar{I}(i,j)$ denote the intensities of I and $\bar{I}$, respectively. The details to calculate SSIM can be found in [14]. Larger PSNR or SSIM values signify better signal restoration.

Table 5 lists the PSNR and SSIM values of the software-oriented bilateral filter, the design in [9], and the proposed design. The quality of the reconstructed images of our design is slightly superior to that of [9]. Fig. 8-10 shows the details of a noisy image and reconstructed images of different designs in restoring $\sigma_{noise} = 20$ image for Lighthouse, restoring $\sigma_{noise} = 30$ image for Boat and restoring $\sigma_{noise} = 40$ for Lena, respectively.

TABLE 5. Average psnr/ssim for test images[1].

Standard deviation	Bilateral filter (Software)	[9] (Hardware)	Proposed design (Hardware)
10	31.428/0.822	32.097/0.833	31.786/0.831
20	28.169/0.721	28.249/0.683	28.295/0.724
30	26.461/0.645	25.916/0.562	26.497/0.643
40	25.236/0.582	24.205/0.472	25.230/0.577
50	24.220/0.530	22.861/0.405	24.193/0.523
60	23.315/0.487	21.764/0.355	23.274/0.478

[1]Test images: Barbara, Boat, Bridge, Couple, Goldhill, Lake, Lena, Lighthouse, Peppers, and Plane

V. CONCLUSION

A low-cost and real-time VLSI architecture of a bilateral filter is proposed in this paper. Distance-oriented grouping, resource sharing, and LUT reduction are applied to reduce the required multipliers and memory space. Experimental results show that the proposed architecture is more cost efficient than [9], [13] while maintaining the same image quality, frame rate, and working clock frequency.

ACKNOWLEDGMENT

All test images in this article are available in the website: http://www.hlevkin.com/06testimages.htm. The authors would like to thank the anonymous reviewers and the editor for their valuable comments and suggestions to improve the quality of the article.

REFERENCES

[1] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," in *Proc. 6th Int. Conf. Comput. Vis.*, Jan. 1998, pp. 839–846.

[2] M. G. Mozerov and J. van de Weijer, "Accurate stereo matching by two-step energy minimization," *IEEE Trans. Image Process.*, vol. 24, no. 3, pp. 1153–1163, Mar. 2015.

[3] Q. Yang, "Hardware-efficient bilateral filtering for stereo matching," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 5, pp. 1026–1032, May 2014.

[4] C.-T. Huang, "Bayesian inference for neighborhood filters with application in denoising," *IEEE Trans. Image Process.*, vol. 24, no. 11, pp. 4299–4311, Nov. 2015.

[5] M. Zhang and B. K. Gunturk, "Multiresolution bilateral filtering for image denoising," *IEEE Trans. Image Process.*, vol. 17, no. 12, pp. 2324–2333, Dec. 2008.

[6] Q. Yang, J. Tang, and N. Ahuja, "Efficient and robust specular highlight removal," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 6, pp. 1304–1311, Jun. 2015.

[7] Q. Yang, S. Wang, and N. Ahuja, "Real-time specular highlight removal using bilateral filtering," in *Proc. Eur. Conf. Comput. Vis.*, 2010, pp. 87–100.

[8] Z. Li, J. Zheng, Z. Zhu, and S. Wu, "Selectively detail-enhanced fusion of differently exposed images with moving objects," *IEEE Trans. Image Process.*, vol. 23, no. 10, pp. 4372–4382, Oct. 2014.

[9] A. Gabiger-Rose, M. Kube, R. Weigel, and R. Rose, "An FPGA-based fully synchronized design of a bilateral filter for real-time image denoising," *IEEE Trans. Ind. Electron.*, vol. 61, no. 8, pp. 4093–4104, Aug. 2014.

[10] H. Dutta, F. Hannig, J. Teich, B. Heigl, and H. Hornegger, "A design methodology for hardware acceleration of adaptive filter algorithms in image processing," in *Proc. IEEE 17th Int. Conf. Appl.-Specific Syst., Archit. Processors (ASAP)*, Sep. 2006, pp. 331–340.

[11] T. Q. Vinh, J. H. Park, Y.-C. Kim, and S. H. Hong, "FPGA implementation of real-time edge-preserving filter for video noise reduction," in *Proc. Int. Conf. Comput. Electr. Eng.*, Dec. 2008, pp. 611–614.

[12] C. Charoensak and F. Sattar, "FPGA design of a real-time implementation of dynamic range compression for improving television picture," in *Proc. 6th Int. Conf. Inf., Commun. Signal Process.*, Dec. 2007, pp. 1–5.

[13] C. S. Chandni and R. Pushpakumari, "Reduced hardware architecture of bilateral filter for real time image denoising," in *Proc. Int. Conf. Intell. Comput., Instrum. Control Technol. (ICICICT)*, Jul. 2017, pp. 769–774.

[14] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Trans. Image Process.*, vol. 13, no. 4, pp. 600–612, Apr. 2004.

CHIH-YUAN LIEN received the B.S. and M.S. degrees in computer science and information engineering from National Taiwan University, Taiwan, in 1996 and 1998, respectively, and the Ph.D. degree in computer science and information engineering from National Cheng Kung University, Taiwan, in 2009. He is currently an Associate Professor with the Department of Electronic Engineering, National Kaohsiung University of Science and Technology, Taiwan. His research interests include image processing, very large scale integration chip design, and video coding systems.

CHI-HUAN TANG received the B.S. and M.S. degrees in electronic engineering from the National Kaohsiung University of Applied Sciences, Kaohsiung, Taiwan, in 2016 and 2018, respectively. He is currently pursuing the Ph.D. degree with the Department of Computer Science and Information Engineering, National Cheng Kung University. His current research interests include artificial intelligence, image processing, and very large scale integration chip design.

PEI-YIN CHEN (Member, IEEE) received the B.S. degree in electrical engineering from National Cheng Kung University, Taiwan, in 1986, the M.S. degree in electrical engineering from Penn State University, Pennsylvania, in 1990, and the Ph.D. degree in electrical engineering from National Cheng Kung University, in 1999. He is currently a Professor with the Department of Computer Science and Information Engineering, National Cheng Kung University. His research interests include very large scale integrated chip design, video compression, fuzzy logic control, and gray prediction.

YAO-TSUNG KUO received the B.S. degree in computer science and information engineering from National Cheng Kung University, Tainan, Taiwan, in 2014, where he is currently pursuing the Ph.D. degree with the Department of Computer Science and Information Engineering. His current research interests include image processing, video compression, and very large scale integration chip design.

YUE-LING DENG received the M.S. degree in computer science and information engineering from National Cheng Kung University, Tainan, Taiwan, in 2016. Her research interests include image processing, very large scale integration chip design, and embedded systems.