Frontiers in Signal Processing, Vol. 4, No. 4, October 2020
https://dx.doi.org/10.22606/fsp.2020.44002

Fast Median Filter Image Processing Algorithm and Its FPGA Implementation

Biaobiao Wang, Qiang Xiang*

College of Electrical & Information Engineering, Southwest Minzu University, Chengdu 610041, China
Email: xqiang_0426@163.com

Abstract. Aiming at the difficulty of real-time and high-speed processing of the filtering process using software methods in image preprocessing, a fast median filtering algorithm based on FPGA is designed using the parallel processing capability of field programmable gate array (FPGA). Compared with the traditional median filtering algorithm, it has a great improvement, reducing the number of data comparisons and improving the processing speed. From the results of the Matlab and Modelsim joint simulation experiments, it can be seen that the filtering algorithm has a good filtering effect and can be fast, Efficiently filter the image, which helps to improve the quality of image processing.

Keywords: field programmable gate array (FPGA), fast median filer, image processing, pipeline.

1 Introduction

In the process of digital image processing, due to factors such as image acquisition tools and methods, various types of noise will inevitably be generated, resulting in blurred and distorted images, thereby reducing image quality and affecting subsequent processing effects [1-3]. Therefore, when processing the image, it is necessary to perform pre-processing such as filtering and enhancement to improve the visual quality of the image. The amount of data processed by the image pre-processing algorithm is very large, and it will cost large of time to implement it with general software, so it is more suitable to use the hardware method to implement it. Field programmable gate array, it is a product of further development on the basis of programmable devices such as PAL, GAL, CPLD, etc. It is flexible in programming and easy to modify, and is very suitable for data processing in pipeline mode and parallel mode. Therefore, it is an ideal choice to use FPGA for image pre-processing.

Median filtering is a non-linear filtering method, which can effectively smooth noise and eliminate impulse interference, has a better filtering effect on salt and pepper noise, and can retain the edge information of the image to a large extent. Therefore, it is widely used in many fields such as digital image smoothing and data analysis and processing [4]. This article uses FPGA to realize the design of a fast median filter, and finally through Modelsim and Verilog language to carry on the simulation verification and compare with the software realization result.

2 The Principle of Image Median Filtering

2.1 Traditional Median Filter

Median filtering is a nonlinear signal processing technology based on statistical ranking theory, which can effectively suppress noise. It was first proposed in the 1970s [5]. The basic principle is to sort the grey levels of pixels in the field of each legal pixel in an image, and then select the middle value of the group as the output pixel value. The traditional median filter is defined as follows:

$$g(x,y) = median\left\{f\left(x-i,y-i\right)\right\},\left(i,j\right) \in S \tag{1}$$

In the above formula, $g(x, y)$, $f(x, y)$ are pixel grey values, and S is the template window. In practical applications, it is commonly used to select a template window with S of 3×3 or 5×5 to process pixels. Sometimes in order to simplify the filter window and increase the running speed, you can also use linear windows, square windows, and cross-shaped windows [6]. The traditional median filter algorithm needs to

sort all pixels in the template window, and then take the median value to output. The process of sorting is the process of comparing and exchanging pixels. The number of comparisons between pixels in a sequence is an important factor affecting the sorting speed. For a filter window of N×N (N takes an odd number) size, the median value needs to be compared $N^2\times(N^2-1)/2$ times for each sorting, and the time complexity is $O(N^2)$. Therefore, although the traditional algorithm is simple, it has a large amount of calculation, a large amount of on-chip resources is consumed for implementation in FPGA, and the processing speed is slow, so it cannot meet the real-time requirements.

2.2 Fast Median Filter

Although the traditional median filter algorithm reduces the number of comparisons for finding the median in the filter window, the number of comparisons is still more. In order to find the median with a lower number of comparisons, this paper adopts a fast median filtering algorithm, which is based on the statistical sorting theory and makes full use of the essence of the sorting algorithm for sorting, which can reduce the filter window to find the median more efficiently. For ease of description, each pixel in the 3×3 filter window is represented as D11, D12, D13, D21, D22, D23, D31, D32, D33, respectively. Table 1 shows the arrangement of pixels in the window.

Table 1. Arrangement of pixels in the window

	Column1	Column2	Column3
Row1	D11	D12	D13
Row2	D21	D22	D23
Row3	D31	D32	D33

When processing, firstly, sorting the data of each row separately to get the maximum, intermediate and minimum values of each row. The maximum value obtained in the first line is: Max1=max {D11, D12, D13}; the middle value is: Med1=med {D11, D12, D13}; the minimum value is: Min1=min {D11, D12, D13}. Similarly, the three values Max2, Med2, Min2 in the second row and the three values Max3, Med3, Min3 in the third row can also be obtained. Since among the above nine numbers, the maximum of the three maximum values must be the maximum of nine pixel values. In the same way, the minimum of the three minimum values must be the minimum of the nine pixel values. The analysis shows that the maximum of the three median values is at least greater than the five pixel values, that is, the minimum value in this row and the median and minimum values in the other two rows. The minimum of the three median values is at least less than five pixel values, that is, the maximum value of this row and the median and minimum values of the other two rows. Therefore, the intermediate value obtained by comparing the minimum value Min_of_max among the three maximum values, the intermediate value Med_of_med among the three median values, and the maximum value Max_of_min among the three minimum values is the final filtering result Med_of_nine. The specific process is to find the minimum of the three maximums: Min_of_max={Max1,Max2,Max3}, the median of the three medians: Med_of_med=med{Med1,Med2,Med3} and the maximum of the three minimums: Max_of_min=max{Min1,Min2,Min3}; Finally, compare the three values obtained above to obtain the final median value: Med_of_nine=med{Min_of_max,Med_of_med,Max_of_min}.

Obviously, compared with the traditional sorting method, the comparison times of this method are greatly reduced. Because only 21 comparison operations were used to find the median. However, using the traditional median method to find the median value requires 36 comparison operations. In comparison, the calculation speed of this algorithm is increased by 42%, so it is very convenient for parallel processing on FPGA.

3 FPGA Realization of Median Filter Algorithm

The realization process of median filter algorithm on FPGA is mainly divided into three parts: 3×3 filter window generation module, rank counter module and median filter algorithm design module. The filter window generation module mainly fetches the digital image pixel data stored in the memory in rows and

generates the 3×3 window required to implement the algorithm. The row and column counter module are mainly used to calculate the row and column values of the entire image, and it can control when the filtering algorithm starts and when it ends. The filtering algorithm module mainly performs median filtering algorithm processing on pixels. The image data is input from datain[7:0], and dataout[7:0] is the filtered output value. clk is the always signal of the system, rst_n is the reset signal, low level is active, and col and row are the row and column positions of the currently processed pixels respectively. The overall design scheme is shown as in Fig. 1.

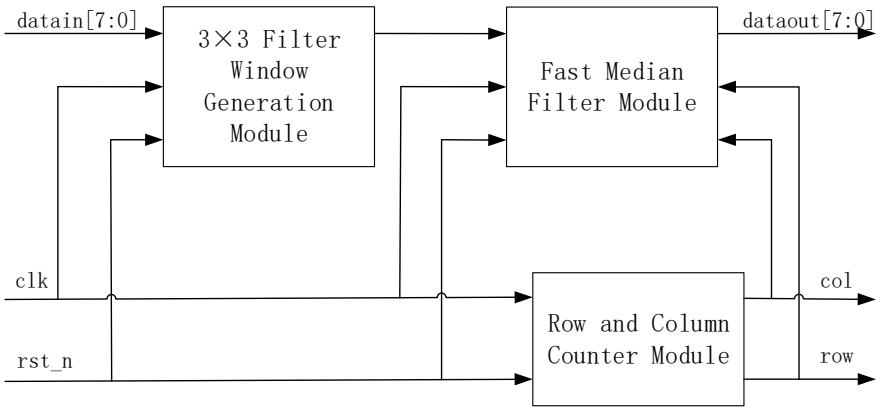

Figure 1. Fast median filter structure

3.1 3×3 Filter Window Generation Module

As shown in Figure 2, the filter window is made up of 9 registers and 2 FIFOs in order to store the rows and columns of data. Each time the FPGA performs noise detection on the image, 256×2+3 data must be buffered before the sampling module can start to obtain the image data in the 3×3 window. Each register stores 1 pixel, and FIFO stores 253 pixels. Through the filter window generation module, image data can be converted from serial input to parallel 3×3 window output, thereby speeding up image processing.

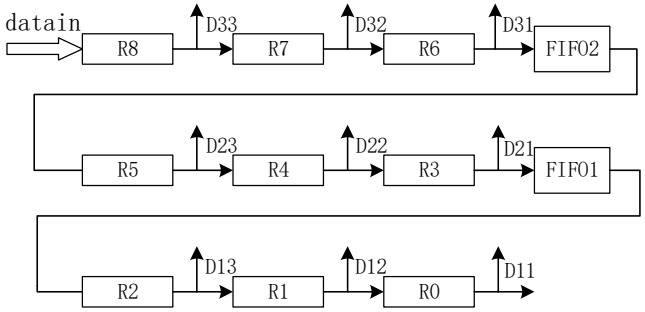

Figure 2. Filter window generation flowchart

The key to realizing a 3×3 window is when to allow two FIFOs to be read and when to allow window data to be valid. When the number of bytes in the FIFO reaches 252, the FIFO is allowed to be read. When 2×256+3=515 clock cycles have passed, the window data is allowed to be valid, and the data stream will continue to enter in the future, and each image data stream will produce a 3×3 window.

3.2 Row and Column Counter Module

When performing median filtering on a two-dimensional image, as the window slides, when the centre pixel is at the edge of the image, part of the data read by the 3×3 filter window is not its area, and the

area data output by the filter window is invalid. Usually, the edge of the image does not contain important information, so the pixels of the edge can be set to zero. The row and column counter module are mainly used to determine the position of the current centre pixel to control the edge. For an M×N image, when the row count value row=1 or row=M, and the column count value col=1 or col=N, the centre pixel of the filter window is located at the edge, and zero is directly output, otherwise the filter processing is performed.

3.3 Median Filtering Algorithm Design

This article uses a 3×3 filter window, as shown in Figure 3. The first level is to use three 3-input comparators to complete the sorting. Put the smallest three numbers together, the middle three numbers together, and the largest three numbers together, to participate in the second comparison. The second level uses three 3-input comparators to find the minimum of the three maximum values, the median of the three medians, and the maximum of the three minimums of the output of the first stage. Use the three output results for the third comparison. The third level uses a three-point comparator to find the median of the three values output by the second stage, which is the final median. The median filtering algorithm is completed.

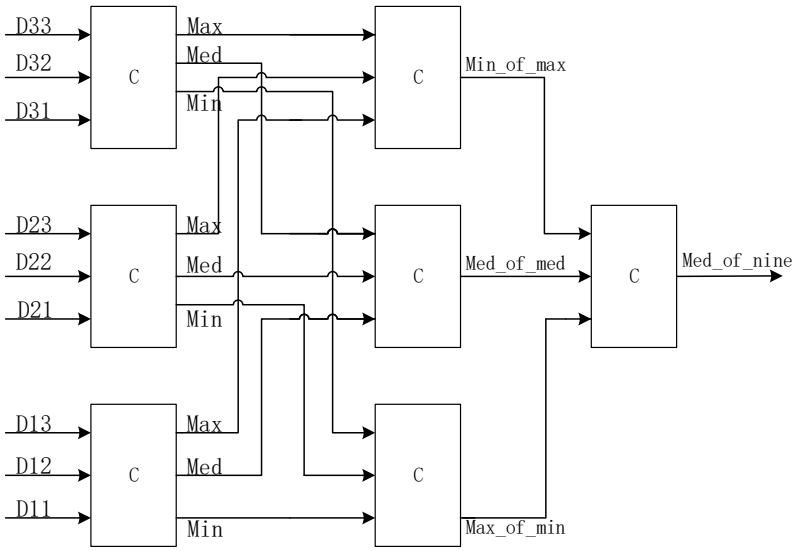

Figure 3. Median filtering algorithm flowchart

4 FPGA Implementation Results

4.1 Simulation of Median Filter Algorithm

The median filter designed in this paper is implemented using XC7Z010CLG400 in the Zynq-7000 series of Xilinx. Use Modelsim and Matlab for simulation, first use Matlab to convert an image with a resolution of 256×256 into a matrix with gray values between 0 and 255[7], and save it in a ".txt" file in hexadecimal format. Then use the gray value matrix as the input signal of the image data to convert it into the test vector file of Modelsim [8]. Finally, after testing the vector file with this design, the display result is shown in Figure 4.

From the figure, we can know that the rst_n signal is valid from the second clock cycle. Since two rows of image data must be read in to generate a 3×3 window, 515 clock cycles are required. It takes 9 clock cycles to complete a 3×3 window to take the median value, so the first median filter result is output in the 525th clock cycle. This pipeline operation and parallel calculation method greatly improves the processing speed of the algorithm, and is very suitable for image pre-processing with high real-time requirements.

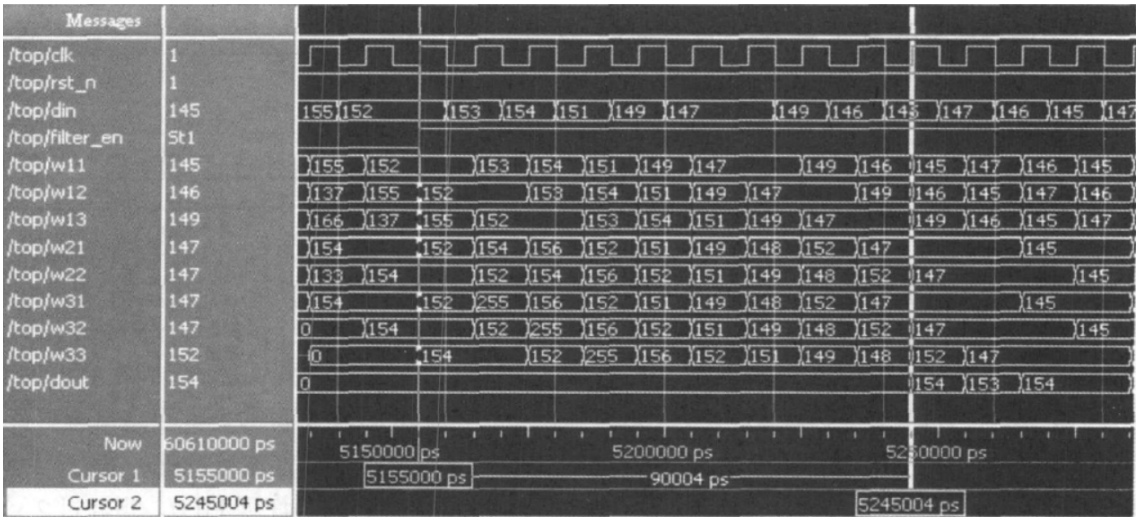

Figure 4. Median filtering algorithm Modelsim simulation diagram

To further verify that the improved median filter used in this paper has higher real-time performance and lower energy consumption than the traditional median filter. By implementing the two algorithms on FPGA respectively, and then comparing their resource utilization on FPGA. The comparison results are shown in Table 2.

Table 2. FPGA resource utilization table

	Combinational functions	Logical registers	Memory bits
Traditional algorithm	325	170	15312
Fast algorithm	152	78	9521

It can be seen from Table 2 that compared to the traditional median filter algorithm, hardware implementation of the fast median filter algorithm will take up less hardware resources, thereby saving valuable chip resources for implementing other processing functions on a single FPGA. In addition, the fast median filter algorithm can achieve a higher operating frequency on the FPGA, thus meeting the requirements of real-time image processing.

4.2 Experimental Results and Discussion

(a) The original image (b) Noisy image

(c) Median filtered image (d) Improve the results of median filtering

Figure 5. Comparison chart of algorithm processing results

The pre-prepared pictures with added salt and pepper noise are processed by Matlab and the median filter algorithm implemented in this paper, and the processed results are shown in Figure 5.

Obviously, the traditional median filter algorithm blurs the image to a certain extent, but the improved median filter algorithm in this paper is closer to the original image and retains more detailed information. Through experimental comparison, the median filter implemented in this paper is more effective than the median filter algorithm implemented by Matlab, and the effect of printing and filtering salt and pepper noise and protecting the edge of the image is better than the results processed by Matlab.

5 Conclusion and Outlook

This design has successfully implemented a fast median filtering algorithm on the Zynq-7000 series FPGA of Xilinx Company, which can effectively filter out the salt and pepper noise in the image, and the real-time performance of the system is well guaranteed. This method provides a solution for the filtering of the front-end pre-processing of the image processing system, and has high application value under the premise of high-speed processing speed and high real-time performance. But at the same time, it was also found that the median filter algorithm caused a certain blur to the edge information in the image. In the future, based on this design, the algorithm will be improved by setting a threshold to determine whether the median is a valid pixel.

Acknowledgments. This work was supported by the Southwest Minzu University Graduate Innovative Research Project (Master Program CX2020SZ96). A special acknowledgement should give to Southwest Minzu University for its experimental conditions and technical support. In addition, Thanks to the careful guidance of the teacher and the help of the classmates.

References

1. Hu Xuelong, Xu Kaiyu. Digital Image Processing. Beijing: Electronic Industry Press, 2014.
2. Chang JJ. Modified 2D median filter for impulse noise suppression in a real-time system. IEEE Trans. on Consumer Electronics, 2005, 41(1):73-80.
3. Wen Qiang, Hou Yongyan. Digital Image Processing. Xi'an: Xidian University Press, 2009.
4. Hu Hongwei. Research on FPGA-based image edge detection system [D]. Harbin Institute of Technology, 2017.
5. Xia Liangzheng. Digital Image Processing [M]. Nanjing: Southeast University Press, 1999:154-160.

6. Wen K L,Burgess N.Listless zerotree coding for color image[C]//Proceedings of the 32nd Asilomar Conference on Signals,System and Computers,1998:231-235.
7. Gonzalez R C, Woods R E, Eddins S L. Digital Image Processing Using Matlab[M]. Ruan Qiuqi, Trans. Beijing: Publishing House of Electronics Industry, 2006.
8. Xu, Q.; Varadarajan, S.; Chakrabarti, C.; Karam, L.J.A distributed canney edge detector: Algorithm and FPGA implementation[J]. IEEE Transactions on Image Processing, 2014, Vol.23, No.7, 2944-2960.

